Parallel stochastic line search methods with feedback for minimizing finite sums

Dragana Bajović * Dušan Jakovetić † Nataša Krejić†
Nataša Krklec Jerinkić †

February 19, 2019

Abstract

We consider unconstrained minimization of a finite sum of N continuously differentiable, not necessarily convex, cost functions. Several gradient-like (and more generally, line search) methods, where the full gradient (the sum of N component costs' gradients) at each iteration k is replaced with an inexpensive approximation based on a sub-sample $\mathcal{N}_k$ of the component costs' gradients, are available in the literature. However, a vast majority of the methods considers pre-determined (either deterministic or random) rules for selecting subsets $\mathcal{N}_k$; these rules are unrelated with the actual progress of the algorithm along iterations. In this paper, we propose a very general framework for nonmonotone line search algorithms with an *adaptive* choice of sub-samples $\mathcal{N}_k$. Specifically, we consider master-worker architectures with one master and N workers, where each worker holds one component function f_i. The master maintains the solution estimate x_k and controls the states of the workers (active or inactive) through a single scalar control parameter p_k. Each active worker sends to the master the value and the gradient of its component cost, while

[1]Department of Power, Electronics and Communication Engineering, Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 2, 21000 Novi Sad, Serbia, e-mail: dbajovic@uns.ac.rs..

[2]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia, e-mail: djakovet@uns.ac.rs, natasak@uns.ac.rs, natasa.krklec@uns.ac.rs. Research supported by Serbian Ministry of Education, Science and Technological Development, grant no. 174030

inactive workers stay idle. Parameter p_k is proportional to the expected (average) number of active workers (which equals the average sample size), and it can increase or decrease along iterations based on a computationally inexpensive estimate of the algorithm progress. Hence, through parameter p_k, the master sends *feedback* to the workers about the desired sample size at the next iteration. For the proposed algorithmic framework, we show that, for any outcome ω (where ω corresponds to one realization of the full run of the algorithm), every accumulation point of sequence $\{x_k\}$ is a stationary point of the full cost function. Under the strong convexity assumption, we provide linear convergence result and the worst-case non-asymptotic analysis. Simulations on both synthetic and real world data sets illustrate the benefits of the proposed framework with respect to the existing non-adaptive rules.

Key words: Variable sample methods; Stochastic optimization; Parallel algorithms; Non-convex cost functions; Feedback.

1 Introduction

We consider problems of the form:

$$\min_{x\in\mathbb{R}^n} f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{1}$$

where each $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., N$, is a continuously differentiable (not necessarily convex), deterministic function bounded from below. Such problems arise frequently in machine learning, where a (usually large scale sized) training data set is partitioned into N subsets, and each f_i represents a loss with respect to each of the training data subsets. (Notice that usually f_i here is itself a sum over the individual training data examples but this is abstracted here.)

In the scenarios of large training sets, it can be beneficial to apply gradient-like methods, where the full gradient (gradient of f) at each iteration k is replaced with an inexpensive estimate. Usually, the estimate of $\nabla f(x_k)$ ($x_k \in \mathbb{R}^n$ being the current iterate) is of the form: $\frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k} \nabla f_i(x_k)$, where $\mathcal{N}_k$ is a subset of $\mathcal{N} = \{1, ..., N\}$. Therefore, in a sense, instead of working with the full cost function f, such algorithms work at each iteration k with a less expensive, but inexact version $f_{\mathcal{N}_k}(x_k) := \frac{1}{|\mathcal{N}_k|}\sum_{i\in\mathcal{N}_k} f_i(x_k)$. The

literature on methods of this type is extensive. We reference here a representative sample of the works; namely, the methods include incremental, e.g., [7, 37, 25, 35], stochastic, e.g., [50, 59, 46, 56, 38, 29, 18, 40], and hybrid methods, e.g., [22, 48, 14, 12]. Besides the gradient approximations, they can also utilize different search directions (generated according to the available information – the available subset $\mathcal{N}_k$), including, e.g., Quasi-Newton-type directions, e.g., [22, 31, 52, 15, 36, 6, 24, 17]. Indeed, in this context, reference [11] demonstrates efficiency of a variable sample size Newton-CG method proposed therein.

Variable sample size approach or adaptive scheduling schemes have also been considered for solving problems of type (1). Reference [26] considers the case of convex stochastic optimization and derives error bounds in terms of sample size. Reference [54] carries out a similar analysis for the strongly convex case. A relationship between the sample size growth and the deterministic convergence rate for a class of optimization methods is considered in [41, 42]. A set of conditions that ensures almost sure convergence is presented in [42], together with a specific recommendation for sample size and error tolerance sequences. Another interesting approach that offers a quantitative measure of quality of a solution is presented in [47]. Therein, optimality functions for general stochastic programs (expected value objective and constraint functions) are considered and an algorithm that utilizes optimality functions to select sample size is developed. An adaptive sample size scheduling is considered in [28] as well. Different types of stochastic equilibrium problems and applications of variable sample size schemes are the topic of study in [55]. Furthermore, variable sample scheduling for second order methods is employed in [10, 11]. More recent relevant works on adaptive sampling and stochastic line searches include, e.g., [12, 13, 43]. In [12], the authors propose a stochastic optimization method that adaptively controls the sample size used in the computation of gradient approximations based on a novel inner product test. In [13], adaptive increase of the sample size is incorporated in the second order (L-BFGS) method. Reference [43] incorporates the backtracking Armijo line-search to the stochastic optimization setting, assuming that the function and gradient values are available up to a dynamically adjusted accuracy. A review of variable sample size methods is available in [32].

In this paper, building from references [2, 3, 4] and the prior work [30, 31], we propose a framework for nonmonotone line search methods with an *adaptive* choice of the sets $\mathcal{N}_k$. We consider the commonly used master-

worker model, e.g., of computing machines in a cluster, with one master node and N worker nodes; see, e.g., [1, 33, 9, 57, 51], for similar models. Each worker i holds one function f_i in (1) and can evaluate this function's values and its gradients. This model is used frequently for large-scale distributed optimization in cluster or cloud environments, whereby the training data set at each worker i – which parameterizes function f_i – is so large that it is infeasible to transfer or store all the workers' data at the master; or, in alterative, the workers cannot send their data to the master due to privacy constraints (while the variable dimension n and the number of nodes N are of a moderate to medium size.)

The master node coordinates computation and updates the solution estimate $x_k \in \mathbb{R}^n$ (see Figure 1). Each worker, at each iteration k, can be in two possible states: active and inactive. Active workers i compute pairs $(f_i(x_k), \nabla f_i(x_k))$ and send them to the master, while inactive workers stay idle. The state of each worker, and therefore the (average) size of the sample $\mathcal{N}_k$, is controlled by the master, and it is increased or decreased as needed, based on the actual progress of the algorithm. Therefore, the algorithm incorporates *feedback information* from the master to the workers about what (average) sample size is needed for efficient progress at the next iteration. The proposed framework considers a nonmonotone line search, and it is flexible with respect to the utilized search directions – we allow for arbitrary directions which are descent with respect to the available function approximation $f_{\mathcal{N}_k}$.

In more details, at each iteration k, the master node broadcasts to all workers the current estimate x_k and a control parameter $p_k \in [0, 1]$. Upon reception of p_k, each worker i decides, independently from other workers but dependently upon its previous state (see Section 2 for details), whether it will be active at iteration k; it becomes active with probability p_k (in which case it evaluates the pair $(f_i(x_k), \nabla f_i(x_k))$ and transmits it to the master) and inactive with probability $1 - p_k$ (in which case it stays idle). Hence, the master works with a sample of an average size $N\,p_k$, updates x_k via a nonmonotone line search rule, and it decides on the value of the next control parameter p_{k+1}; the latter quantity can either decrease, stay equal, or increase with respect to p_k. The update rule for the control parameter p_k is based on the comparative size of two quantities, which we denote by dm_k and ε_k. The quantity dm_k is a suitable measure of progress made with respect to the current function $f_{\mathcal{N}_k}$; the quantity ε_k estimates how different $f_{\mathcal{N}_k}$ is with respect to the true objective f at the current iterate x_k. The

adaptive rule for choosing p_k operates as follows. If dm_k is small relative to ε_k, p_{k+1} increases with respect to p_k. Intuitively, the progress which is possible based on the average sample size equal to $N\,p_k$ is exhausted, and hence the precision of approximating f should be increased. Conversely, if dm_k is large with respect to ε_k, then p_{k+1} is decreased with respect to p_k - the progress can still be achieved even with a lower precision, and hence the average sample size is decreased. The change (increase or decrease) in p_k is set such that the quantities dm_k and ε_k are kept in a balance (i.e., they have comparable values) across iterations.

Our main results are as follows. Assuming that the f_i's are continuously differentiable and bounded from below (and not necessarily convex), we show that, eventually (starting from a random, i.e., outcome dependent, but finite $\bar{k}$) $p_k = 1$ for all $k \geq \bar{k}$, i.e., the master works with the full sample. This eventual involvement of all workers in the optimization process, i.e., the achievement of the "full precision," is not artificially enforced in the algorithm, but is rather a result of a carefully designed feedback process between the master and the workers. Moreover, we show that every accumulation point of the sequence of iterates $\{x_k\}$ is a stationary point of the desired cost function f. We also provide some non-asymptotic analysis where the worst-case complexity is stated for strongly convex case. The proposed method incurs significant communication and computational cost savings on the simulated problem formulations and instances, when compared with the methods where the true objective function (and its gradient) is used at all iterations, and with the hybrid (incremental) scheme in [22].

Therefore, the purpose of the current paper is to introduce the proposed adaptive framework, establish convergence guarantees to a stationary point under very generic, not necessarily convex, costs, and deliver initial numerical results.

This paper builds on references [2, 3, 4] and [30, 31] for centralized optimization. In [2, 3, 4], the authors propose an algorithm which allows that the sample size both increase and decrease along iterations within the trust region framework. A similar mechanism for the schedule sequence in a line search framework is developed in [30]. Reference [31] extends the work in [30] to a nonmonotone line search framework. In this paper, we also consider a nonmonotone line search framework, as done in [31]. However, a major difference of the current paper with respect to [31] and [2, 3, 4, 30] is that here the sample size is controlled only in terms of its mean value, and not in terms of the actual current size. From the implementation perspective, the frame-

work proposed here is much more suitable for, e.g., cluster environments, as the master controls the current sample through a single scalar parameter p_k, which it broadcasts to all workers. In contrast, with the algorithms in [2, 3, 4, 30, 31], the master needs a more complex control mechanism, for example to contact each of the workers individually and declare each of them active or inactive. Finally, from the perspective of the algorithm design and analysis, the simplified sample size control here corresponds to a more challenging situation in the design of the control rules and the algorithm analysis. Specifically, compared with [31] – the work closest to this paper – we introduce here a very different lower bound for controlling the p_k's which is essential to ensure eventual activation of all workers and hence the convergence to a stationary point.

Paper organization. Section 2 describes the model that we assume and presents the proposed algorithmic framework, while Section 3 provides its convergence analysis. In Section 4 we present the results of the initial numerical testings. Finally, in Section 5, some conclusions about the proposed framework are drawn.

Most of the scalar and vector quantities in the sequel are random, i.e., dependent on an outcome ω of the underlying probability space. (Throughout, ω corresponds to a realization of the full algorithm run.) Where needed, we will explicitly state whether a certain quantity is deterministic or random and omit ω wherever there is no possibility of confusion.

2 Model and algorithm

We consider a master-worker computational model (Figure 1); see also [1, 33, 9, 57, 51]. Each worker i holds a continuously differentiable, not necessarily convex, function $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., N$. Unlike, e.g., [22], we do not impose any assumptions on the "similarity" among the f_i's; that is, we do not require that the minimizers of the individual f_i's – if they exist – are close or within a pre-defined distance from each other. This allows, for example, that the training data sets from different workers may be generated from very different distributions. We state the formal standard assumption below.

A 1. *Each function $f_i : \mathbb{R}^n \to \mathbb{R}$, $i \in \mathcal{N}$, is continuously differentiable and bounded from below.*

Notice that Assumption A1 implies that the objective function f is also

continuously differentiable and bounded from below.

The master-worker system solves problem (1) through an iterative algorithm, as follows. The algorithm consists of the outer iterations k and the inner iterations s. We first describe the outer iterations k.[1] The master node maintains over k the solution estimate $x_k \in \mathbb{R}^n$. At each iteration k, the master node broadcasts to all workers quantity x_k and a scalar control parameter $p_k \in [0, 1]$. At each iteration k, each worker can be in one of the two possible states: active and inactive. As we will see further ahead, the state of each worker does not change along the *inner* iterations. Each active worker i calculates the pair $(f_i(x_k), \nabla f_i(x_k))$ and sends it back to the master. Notice that the goal of outer iterations is to get $\nabla f_{\mathcal{N}_k}(x_k)$, and the goal of the inner iterations is to get α_k.

The assumed abstract master-worker architecture is well suited for situations where the decision variable dimension n and the number of nodes N are of a moderate to medium size, while each f_i may correspond to a large-sized training data set, so that it is economical that each worker stores and keeps locally this training data, calculates locally the pair $(f_i(x_k), \nabla f_i(x_k))$, and then communicates the pair with the master. Notice that the physical implementation of the master node here can correspond to multiple actual machines with shared parameters. See, e.g., [34], for actual physical implementations of architectures similar to the simplified, abstract architecture considered here.

The parameter p_k plays the role of activation probability of each worker at iteration k. Upon reception of p_k, each worker decides, independently from other workers, whether it will be active or inactive at iteration k. It becomes active with probability p_k and inactive with probability $1-p_k$. More precisely, the state update of each worker is performed as follows, depending on the current and previous values of the control parameter, p_k and p_{k-1}. (Notice that each worker i stores and memorizes both p_k and p_{k-1}.) If $p_k = p_{k-1}$, then each worker i keeps the same state as in the previous iteration $k-1$. Else, if $p_k \neq p_{k-1}$, each worker sets its state at k to active with probability p_k (and to inactive with probability $1 - p_k$), independently from all other workers, and independently from its own and others' previous states. In other words, whenever the control parameter p_k changes, the system is being reconfigured. The mechanism of the update of p_k that we propose is explained later; as we will see, p_k will actually not be a deterministic quantity, i.e., it will be

[1] From now on, we refer to the outer iterations simply as iterations.

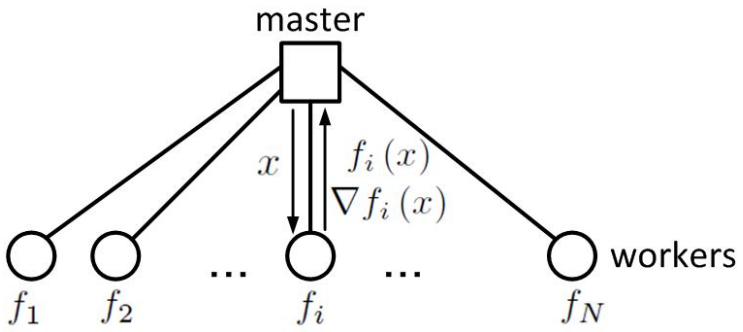

Figure 1: Illustration of the adopted master-worker computational model. At the outer iterations k, the master sends to the workers x_k, while the active workers send back to the master the pair $(f_i(x), \nabla f_i(x))$ where $x = x_k$; at the inner iterations s which occur at the beginning of the k-th outer iteration, the master sends to the workers the trial point $x_k + \beta^s d_k$, while the active workers send back to the master $f_i(x)$ where $x = x_k + \beta^s d_k$.

a random variable. Hence, iterates x_k are also random. Denote by $\tilde{N}_k$ the random number of active workers at iteration k, i.e., $\tilde{N}_k = |\mathcal{N}_k|$. Notice that $\mathcal{N}_k$ is also random. Notice that, if $p_k \neq p_{k-1}$, $\tilde{N}_k$ is a random variable, which, conditioned on the value of p_k, has a Binomial distribution $\tilde{N}_k : \mathcal{B}(N, p_k)$. Otherwise, $\tilde{N}_k$ is determined by the previous number of working nodes, more precisely $\tilde{N}_k = |\mathcal{N}_{k-1}|$. Denote also by $N_k := p_k N$; that is, N_k is the average number of active workers, conditioned on the current value of p_k. In the sequel, ω represents one realization of the infinite sequence of sets $\mathcal{N}_k(\omega)$, $k = 0, 1, ...$

We now explain how the master node updates the solution estimate x_k based on the received information from active workers $(f_i(x_k), \nabla f_i(x_k))$, $i \in \mathcal{N}_k$, i.e., based on the function

$$f_{\mathcal{N}_k}(x) = \frac{1}{|\mathcal{N}_k|} \sum_{i \in \mathcal{N}_k} f_i(x).$$

Notice that, for each fixed constant $x \in \mathbb{R}^n$, $f_{\mathcal{N}_k}(x)$ is random due to the dependence on the random set $\mathcal{N}_k$. Specifically, the subsequent solution estimate is obtained through a nonmonotone line search rule with generic descent directions, as follows:

$$x_{k+1} = x_k + \alpha_k d_k,$$

where d_k is the search direction and α_k is the step size. Due to the randomness of x_k and the fact that step size α_k is the result of a line search dependent on x_k (see the details ahead in Algorithm 3), both α_k and d_k are also random quantities. We allow for a generic descent search direction d_k for function $f_{\mathcal{N}_k}$, i.e., d_k is such that, for every outcome ω, there holds:

$$d_k^T \nabla f_{\mathcal{N}_k}(x_k) < 0.$$

We now discuss some practical possibilities for the choice of d_k. The obvious choice is $d_k = -\nabla f_{\mathcal{N}_k}(x_k)$. However, incorporating some second order information can improve the algorithm speed of convergence significantly, especially if the nonmonotone line search rule is applied as suggested below. For example, one can employ the spectral gradient direction of the form $d_k = -z_k \nabla f_{\mathcal{N}_k}(x_k)$, where z_k is a positive constant obtained trough $x_k - x_{k-1}$ and $\nabla f_{\mathcal{N}_k}(x_k) - \nabla f_{\mathcal{N}_{k-1}}(x_{k-1})$, see, e.g., [8] for details. In this case, the master node has to store the previous solution estimate x_{k-1} and the previous gradient $\nabla f_{\mathcal{N}_{k-1}}(x_{k-1})$ in order to calculate the new search direction; that is, it additionally stores two vectors of length n. Alternatively, one can use Quasi-Newton types of directions such as BFGS. Such directions are of the form $d_k = -H_k \nabla f_{\mathcal{N}_k}(x_k)$, where H_k is updated via an inexpensive recursion, such that positive definiteness of H_k is maintained. Overall, compared with the spectral gradient method, the master additionally stores an $n \times n$ matrix and performs its update over iterations k. Finally, when the memory requirements for BFGS are an issue, one can consider the limited memory BFGS approach as suggested in [10, 11].

We now describe the inner iterations s; their purpose is to determine the step size α_k via an Armijo-type rule. Specifically, once the master calculates d_k, it initiates the inner iterations $s = 0, 1, ...$ At each s, the master evaluates a trial point $x_k + \beta^s d_k$ ($\beta \in (0, 1)$) and sends it to all the workers. Here, β is a deterministic parameter. The workers do not change their state during the inner iterations, i.e., during all inner iterations they stay in their current state. Upon reception of the trial point $x_k + \beta^s d_k$, each active worker i, $i \in \mathcal{N}_k$, calculates the function value $f_i(x_k + \beta^s d_k)$ and sends it back to the master. The process is repeated until the following condition is met for a certain inner iteration s_k (here $\alpha_k := \beta^{s_k}$):

$$f_{\mathcal{N}_k}(x_k + \alpha_k d_k) \leq f_{\mathcal{N}_k}(x_k) + \eta \alpha_k d_k^T \nabla f_{\mathcal{N}_k}(x_k) + e_k, \tag{2}$$

where $\eta \in (0, 1)$ is the deterministic parameter, and e_k represents the so called

"nonmonotonicity" parameter. The sequence $\{e_k\}_{k\in\mathbb{N}_0}$ is deterministic, fixed in advance and it satisfies the following condition.

C 1. *The sequence* $\{e_k\}_{k\in\mathbb{N}_0}$ *is positive and summable, that is*

$$e_k > 0, \ \sum_{k=0}^{\infty} e_k < \tilde{e} < \infty. \tag{3}$$

In general, this nonmonotone line search with backtracking allows for longer step sizes when compared with the Armijo rule (wherein $e_k = 0$). In other words, less trial points are needed and we expect less function evaluations $f_{\mathcal{N}_k}(x_k + \alpha d_k)$ to be calculated. Besides this rule – which originates in [23] – other nonmonotone schemes like, e.g., [58], [20] [27], can also be considered, since the convergence results hold with technical differences in the proofs. The results obtained in a similar variable sample scheme framework [31] indicate that the best choice for the line search rule depends on a search direction to be used. For instance, the negative gradient search direction seems to work better with less freedom in (2), so the parameter e_k for this direction should be modest. Moreover, since nonmonotone line search may be beneficial for deterministic optimization (i.e., for the full sample in our context), especially if the spectral gradient-type direction is used (see [8] for instance), we allow $e_k >$ even if $N_k = N$. However, notice that convergence analysis remains the same if we allow $e_k \geq 0$ instead of $e_k > 0$.

The proposed line search (see step S4 in Algorithm 3) is well defined, i.e., it terminates after a finite number of inner iterations if the function $f_{\mathcal{N}_k}$ is continuously differentiable and bounded from below, as assumed throughout this paper. Moreover, backtracking may be considered as a substitute for Wolfe conditions since it makes the step size not too small [39].

We now explain how the master updates the control parameter and sets the new value p_{k+1}. The update is governed by the actual progress of the algorithm estimated by computationally inexpensive means – this is the feature of our work which distinguishes it from most of the literature. Specifically, the control parameter is updated by comparing two measures of progress: dm_k and ε_k. The quantities dm_k and ε_k, as explained ahead, are random, due to their dependence on x_k. The first measures the progress in decreasing the objective function and is defined by

$$dm_k = -\alpha_k d_k^T \nabla f_{\mathcal{N}_k}(x_k). \tag{4}$$

Some other choices of $dm_k \geq 0$ are possible as well (see [31] for details) but are not considered in this paper. The purpose of the second measure of progress ε_k, is to estimate the error which is generated by employing the sample average function $f_{\mathcal{N}_k}$ instead of f. The quantity $\varepsilon_k = \varepsilon_k(p)$ is a function of the next control parameter $p = p_{k+1}$ which is to be determined; $\varepsilon_k(p)$ may also depend on the data currently available at the master, e.g., on the value $f_{\mathcal{N}_k}(x_k)$ - this is encoded in the subscript k of notation $\varepsilon_k(p)$. This means that for each fixed p, quantity $\varepsilon_k(p)$ is random and depends on ω. As it can be seen from the proofs, the only technical requirement is that $\varepsilon_k(p) \geq 0$ for all $p \in [0, 1]$ and bounded away from zero for all $p \in [0, 1)$. However, a desirable property is that $\varepsilon_k(p)$ is a decreasing function with respect to p. We assume that the control parameter p_k takes values from the discrete set $\Pi = \{\pi_1, \pi_2, ..., \pi_m\}$ fixed in advance, where $0 < \pi_1 < \pi_2 < ... < \pi_m = 1$ are deterministic quantities. The number of allowed values (m) can be arbitrary large. However, our tests indicate that $m \leq N$ provides sufficiently good performance. Before we discus the possible choices of ε_k, we state the condition which is the only one that has to be formally satisfied.

C 2. *For every outcome ω, there holds: $\varepsilon_k(p) \geq 0$, for all $p \in \Pi$, and $\varepsilon_k(p) \geq \kappa > 0$ for all $p \in \Pi \backslash \{1\}$.*

Let us now discuss the specific choices $\varepsilon_k(p)$. One possibility is to set $\varepsilon_k(p) = 1 - p$, which is feasible as we assume that p can take only finitely many discrete values. This is a simple choice which is easy to calculate, but it does not account for the influence of the current solution estimate x_k. In order to incorporate the latter information, one can define

$$\varepsilon_k(p) = \mu \frac{\sigma_k}{\sqrt{pN}}, \tag{5}$$

where μ is a positive deterministic quantity and σ_k^2 measures the variance (spread) of the set of values $f_i(x_k),\ i \in \mathcal{N}_k$ and is defined by

$$\sigma_k^2 = \frac{1}{|\mathcal{N}_k|} \sum_{i \in \mathcal{N}_k} \left(f_i(x_k) - \frac{1}{|\mathcal{N}_k|} f_{\mathcal{N}_k}(x_k) \right)^2. \tag{6}$$

Notice that σ_k^2 is random, e.g., due to its dependence on x_k. Even in the case where the sample variance estimate in (5) is employed, C2 is not too restrictive since we do not expect the variance to converge to zero. However, a positive constant κ can be added to ε_k in (5) as a safeguard, if necessary.

Regarding the update of the control parameter, the main idea is to construct an algorithm that allows us to work with as few nodes as we can and still ensure continuous progress over iterations k, especially at the beginning of the optimization process. However, in order to ensure almost sure convergence towards a stationary point of the true objective function f, we need to reach the whole set of nodes eventually. There are two crucial points which make this possible. One of them is incorporated in step S3 of the main algorithm (see ahead Algorithm 3), and it states that the set of active nodes does not change if the control parameter remains the same. As it can be seen from the proofs, this requirement discards the scenario where $p_k = \overline{p} < 1$ after some finite iteration $\bar{k}$. On the other hand, persistent changes of p_k (i.e., changes of p_k that do not stop after a certain finite iteration) and the subset of active nodes have to be avoided. This is achieved by a random safeguard sequence denoted by $\{p_k^{\min}\}$. This sequence will be a lower bound for the true control parameter p_k, i.e., it will be required that $p_{k+1} \geq p_k^{\min}$ for all k. The update of this safeguard sequence will be described later on.

Let the lowest allowed probability π_1 define the initial safeguard $p_0^{min} := \pi_1$ and take this value as the initial probability, i.e. $p_0 := p_0^{min}$. In order to perform the update of the safeguard p_k^{min}, the master has to store and update a vector of length m which is denoted by $F_k = ([F_k]_1, ..., [F_k]_m)$. The role of this vector, is to track the approximate objective function values for different values of p_k, i.e., for different precision levels. As can be seen in Algorithm 1 ahead, for $k \geq 1$, quantities p_k^{min} and F_k are random. The jth component $[F_k]_j$ corresponds to the precision level controlled by π_j and represents the lowest value of the approximate objective function defined by π_j precision level achieved within the first k iterations. Recall that the control parameter determines only the expected number of working nodes. Therefore, F_k tracks different levels of the expected precision. Each component of this vector is initially set to $+\infty$. It is worth noting that our theory allows that m can be taken independently from N. Hence, when N grows very large, m can be chosen so that it stays bounded as N grows; therefore, storing and maintaining F_k does not induce significant costs.

The algorithm for updating the lower bound p_k^{min} is presented below. The main idea is to increase the lower bound, and therefore increase the expected precision, if there was not enough decrease in the approximate objective function $f_{\mathcal{N}_k}$. More precisely, we track the decrease of the expected precision level determined by p_{k+1} which is to be used in the following iteration. Before stating the algorithm, we introduce parameters θ_k and γ_k which determine

sufficient decrease in the approximate objective function and increase in the lower bound, respectively. They are both generic parameters assumed only to be positive and uniformly bounded away from zero which is formally stated in the following condition. These parameters are stated as random to allow more freedom in their choice, but one can also set them as deterministic sequences fixed in advance.

C 3. *Parameters θ_k and γ_k are such that $\theta_k \geq \theta > 0$ and $\gamma_k \geq \gamma > 0$.*

One possible choice is $\theta_k = p_k$. In that case, $\theta = \pi_1$. Thus, although θ_k is truly random in this case, the condition C3 is easily satisfied with θ and γ being deterministic and fixed in advance. Further details on suitable choices of these parameters are given in Section 4, and the algorithms below are stated with generic values.

The updating of the lower bound p_k^{min} at the master is stated within Algorithm 1 below. Notice that the sequence $\{p_k^{min}\}_{k\in\mathbb{N}}$ is nondecreasing and bounded from above by 1.

ALGORITHM 1.

Updating the lower bound p_k^{min} at the master

S0 Inputs: p_k, p_{k+1}, $f_{\mathcal{N}_{k+1}}(x_{k+1})$, F_k; θ_k, γ_k. Let j be such that $\pi_j = p_{k+1}$.

S1 Updating the lower bound.

If $p_k < p_{k+1}$ and $[F_k]_j - f_{\mathcal{N}_{k+1}}(x_{k+1}) < \theta_k$ set

$$p_{k+1}^{min} = \min\{1, p_k^{min} + \gamma_k\}.$$

Else, $p_{k+1}^{min} = p_k^{min}$.

S2 Updating vector F_k

If $p_k < p_{k+1}$, set

$$[F_{k+1}]_j = \min\{f_{\mathcal{N}_{k+1}}(x_{k+1}), [F_k]_j\}.$$

Else, $F_{k+1} = F_k$.

Notice that if p_{k+1} is used for the first time then $[F_{k+1}]_j = \infty$ and the lower bound remains unchanged. Also, recall that the number of active nodes $|\mathcal{N}_{k+1}|$ does not have to be equal to N_{k+1}.

Now, let us state the algorithm for updating the control parameter p_k and the main algorithm. These algorithms are presented with generic search directions, generic function ε_k and generic parameters θ_k, γ_k. The algorithms allow a lot of freedom in choosing these quantities. In Section 4 we state the specific parameter values used in the initial numerical testings with the BFGS search direction. The convergence (for an arbitrary outcome ω) is proved in Section 3 under rather general assumptions. Furthermore, our initial tests indicate that the safeguard lower bounds $p_k^{\min}$ do not significantly affect the actual update of the control parameter p_k and thus their role is just to provide a theoretical safeguard needed for the convergence proofs.

ALGORITHM 2.

Updating the control parameter p_k at the master

S0 Inputs: $dm_k,\ p_k^{min},\ p_k,\ \varepsilon_k(p_k),\ \nu_1 \in (0,1)$.

S1 If $p_k \leq p_k^{min}$ set $p_{k+1} = \min\{\pi_j \in \Pi : \pi_j \geq p_k^{min}\}$. Else, go to step S2.

S2 **1)** If $dm_k = \varepsilon_k(p_k)$ set $p_{k+1} = p_k$.

2) If $dm_k > \varepsilon_k(p_k)$ set $p_{k+1} = \max\{\pi_j \in \Pi : p_k^{min} \leq \pi_j \leq p_k, dm_k \leq \varepsilon_k(\pi_j)\}$. If such p_{k+1} does not exist, set $p_{k+1} = \min\{\pi_j \in \Pi : \pi_j \geq p_k^{min}\}$.

3) If $dm_k < \varepsilon_k(p_k)$ and

i) $dm_k \geq \nu_1 \varepsilon_k(p_k)$ set $p_{k+1} = \min\{\pi_j \in \Pi : \pi_j \geq p_k, dm_k \geq \varepsilon_k(\pi_j)\}$. If such p_{k+1} does not exist set $p_{k+1} = 1$.

ii) $dm_k < \nu_1 \varepsilon_k(p_k)$ set $p_{k+1} = 1$.

The main idea of Algorithm 2 is to keep the two measures of precision close to each other by adapting the expected precision level. If $dm_k > \varepsilon_k(p_k)$, the precision is too high and hence we decrease it by decreasing the control parameter, if the lower bound p_k^{min} allows. Roughly speaking, the solution is still far away and we want to approach it with smaller cost, i.e., with a smaller number of workers activated. On the other hand, if the measure of progress in the objective function dm_k is relatively small, we increase the expected

precision by increasing the control parameter. The proposed algorithms aim to activate the whole set of nodes eventually, but only when the solution neighborhood is approached.

The proposed overall main algorithm is given in Algorithm 3. Notice that algorithm Algorithm 3 operates in such a way that each worker updates the step length in coordination with the other workers, the master being in charge of computing the search direction and updating the step length at each inner iteration of the line search.

ALGORITHM 3.

The main algorithm

S0 Inputs (the master): $p_0^{min} \in (0,1),\ x_0 \in \mathbb{R}^n,\ \beta, \nu_1, \eta \in (0,1),\ \{e_k\}_{k\in\mathbb{N}_0}$ satisfying C1.

S1 The master sets $p_0 = p_0^{min}$ and sends x_0 and p_0 to all the workers.

S2 Each worker activates with probability p_0, independently from other workers. Each active worker i sends the pair $(f_i(x_0),\ \nabla f_i(x_0))$ to the master. The master sets $k = 0$

S3 The master calculates the search direction d_k such that $d_k^T \nabla f_{\mathcal{N}_k}(x_k) < 0$.

S4 Nonmonotone line search through inner iterations: Find the smallest nonnegative integer s_k such that $\alpha_k = \beta^{s_k}$ satisfies

$$f_{\mathcal{N}_k}(x_k + \alpha_k d_k) \leq f_{\mathcal{N}_k}(x_k) + \eta \alpha_k d_k^T \nabla f_{\mathcal{N}_k}(x_k) + e_k.$$

At each $s = 0, 1, ..., s_k$, the master sends $x_k + \beta^s d_k$ to all active workers, and each active worker sends back $f_i(x_k + \beta^s d_k)$ to the master.

S5 The master sets $x_{k+1} = x_k + \alpha_k d_k$ and $dm_k = -\alpha_k d_k^T \nabla f_{\mathcal{N}_k}(x_k)$.

S6 The master determines p_{k+1} via Algorithm 2 and sends p_{k+1} and x_{k+1} to all the workers.

S7 Workers state update: If $p_{k+1} = p_k$, each worker stays in its previous state, i.e., $\mathcal{N}_{k+1} = \mathcal{N}_k$. Else, each worker activates with probability p_{k+1}, independently from other workers, and independently from its previous states and the previous states of other workers. The set $\mathcal{N}_{k+1}$ consists of the current active nodes.

S8 Each active worker i sends the pair $(f_i(x_{k+1}), \nabla f_i(x_{k+1}))$ to the master.

S9 The master determines p_{k+1}^{min} via Algorithm 1.

S10 The master sets $k = k + 1$. If $\mathcal{N}_k = \mathcal{N}$ and $\nabla f_{\mathcal{N}_k}(x_k) = 0$, stop. Else go to step S3.

Notice from step S10 that we assumed that the master is aware of the total number of workers N. The stoping criterion is as in deterministic optimization - $\nabla f(x_k) = 0$. Notice that the full gradient is calculated only if $\mathcal{N}_k = \mathcal{N}$ so there are no additional costs in step S10.

3 Convergence analysis

This section provides convergence analysis of the proposed algorithm. The first part is devoted to asymptotic result where we prove that the algorithm converges to a stationary point of the considered objective function almost surely, provided that the search direction is descent with respect to the current estimate of the objective function. This result is obtained without imposing convexity and leans on the fact that the full sample is reached eventually. The second part states conditions under which linear convergence with respect to the expected optimality gap is attained. This furthermore implies complexity result stated in the second subsection and provides some non-asymptotic analysis. This result is stated under the strong convexity assumption.

3.1 Asymptotic analysis

In this subsection, we prove that Algorithm 3 converges to a stationary point of (1) for any outcome ω. Therefore, in all the proofs in the sequel work we assume that ω is arbitrary, but fixed and we write ω explicitly in the statements.

In the following theorem we prove that control parameter p_k eventually reaches 1. This is the crucial result for convergence analysis. More precisely, given an arbitrary outcome ω, control parameter $p_k = 1$ for all $k \geq \bar{k}$, where $\bar{k}$ is outcome-dependent, but finite number of iterations. It is important to notice here that $p_k = 1$ might occur at some $k < \bar{k}$ and then decrease to some smaller value, but eventually, i.e., from iteration $\bar{k}$ and onwards,

we have $p_k = 1$. The proof contains two main arguments. First, we show that p_k cannot remain at a constant value strictly smaller than 1 from a certain iteration onwards. Second, we show that p_k cannot exhibit persistent changes, and eventually reaches the value of 1 without further decreases. Regarding the first argument, the proof given here is along the lines of the proof of Lemma 4.1 in [30]. Regarding the second argument, the stochastic nature of the activation sets $\mathcal{N}_k$ here requires a different analysis with respect to [30]. Notice that a different update of lower bounds p_k^{min} is used here. This second argument is the main technical contribution of Theorem 3.1.

Theorem 3.1. *Let A1 and C1-C3 hold and assume that $\{x_k(\omega)\}$ is a sequence of random vectors generated with Algorithm 3. Then, for any fixed outcome ω, the sequence $\{x_k(\omega)\}$ is either finite with $n_1(\omega)$ elements and $x_{n_1}(\omega)$ is the stationary point of the objective function f, or $\{x_k(\omega)\}$ is infinite and there exists finite $\bar{k}(\omega) \in \mathbb{N}$ such that $p_k(\omega) = 1$ for every $k \geq \bar{k}(\omega)$.*

Remark. The proof below works in such a way that it fixes an arbitrary outcome ω, and then it considers the evolution of the relevant quantities $p_k(\omega)$ and $\{x_k(\omega)\}$. All the auxiliary quantities that appear in the proof below, namely all subsequences $K_i(\omega)$, the corresponding values $n_i(\omega)$, and all the values depending on these, such as $\bar{p}(\omega)$ and $r(\omega)$, are also random, i.e., outcome-dependent. Once we fix ω, all the steps in the proof hold surely within this outcome.

Proof. Let us fix an arbitrary outcome ω. According to step S10, the main algorithm terminates only if $\nabla f(x_k) = 0$. Therefore, we consider the case where the number of iterations is infinite. First, we prove that p_k cannot be constant if it is strictly smaller than 1.

Suppose that there exists n_2 such that for every $k \geq n_2$

$$p_k = p^1 \in \Pi \backslash \{1\}.$$

In that case, step S7 of Algorithm 3 implies that $\mathcal{N}_k = \mathcal{N}^1$ for every $k \geq n_2$ for some fixed set $\mathcal{N}^1$. Denoting $g_k^{\mathcal{N}_k} = \nabla f_{\mathcal{N}_k}(x_k)$, we know that for every $k \geq n_2$

$$f_{\mathcal{N}^1}(x_{k+1}) \leq f_{\mathcal{N}^1}(x_k) + \eta \alpha_k (g_k^{\mathcal{N}^1})^T d_k + e_k,$$

i.e., for every $q \in \mathbb{N}$

$$\begin{aligned} f_{\mathcal{N}^1}(x_{n_2+q}) &\leq f_{\mathcal{N}^1}(x_{n_2+q-1}) + \eta\alpha_{n_2+q-1}(g^{\mathcal{N}^1}_{n_2+q-1})^T d_{n_2+q-1} \\ &+ e_{n_2+q-1} \leq \ldots \leq f_{\mathcal{N}^1}(x_{n_2}) \\ &+ \eta \sum_{j=0}^{q-1} \left(\alpha_{n_2+j}(g^{\mathcal{N}^1}_{n_2+j})^T d_{n_2+j} + e_{n_2+j} \right). \end{aligned} \tag{7}$$

Now, assumption A1 implies the existence of a constant M_F such that

$$\begin{aligned} -\eta \sum_{j=0}^{q-1} \alpha_{n_2+j}(g^{\mathcal{N}^1}_{n_2+j})^T d_{n_2+j} &\leq f_{\mathcal{N}^1}(x_{n_2}) - f_{\mathcal{N}^1}(x_{n_2+q}) \\ &+ \sum_{j=0}^{q-1} e_{n_2+j} \\ &\leq f_{\mathcal{N}^1}(x_{n_2}) - M_F + \tilde{e}. \end{aligned} \tag{8}$$

The inequality (8) is true for every q so

$$0 \leq \sum_{j=0}^{\infty} -\alpha_{n_2+j}(g^{\mathcal{N}^1}_{n_2+j})^T d_{n_2+j} \leq \frac{f_{\mathcal{N}^1}(x_{n_2}) - M_F + \tilde{e}}{\eta} := C.$$

Therefore

$$\lim_{k\to\infty} dm_k = \lim_{j\to\infty} -\alpha_{n_2+j}(\nabla f_{\mathcal{N}^1}(x_{n_2+j}))^T d_{n_2+j} = 0. \tag{9}$$

Since $\nu_1\varepsilon_k(p_k) \geq \nu_1\kappa > 0$ for every $k \geq n_2$, there exists $n_3 > n_2$ such that $dm_{n_3} < \nu_1\varepsilon_{n_3}(p_{n_3})$ and therefore Algorithm 2 implies $p_{n_3+1} = 1$ which is in contradiction with the current assumption.

We have just proved that the control parameter cannot stay on $p^1 < 1$. If the lower bound of control parameter p_k^{min} achieves 1 at some finite iteration, the statement of the theorem obviously holds since we have that $p_{k+1} \geq p_k^{min}$ for every k. Now, let us assume that the statement of the theorem is not true and consider the remaining case, i.e., assume that $p_k^{min} < 1$ for all k. This assumption implies that p_k^{min} is increased only at finitely many iterations. Therefore, based on Algorithm 1, we conclude that there exists an iteration n_5 such that for every $k \geq n_5$ we have one of the following possibilities:

M1 $p_{k+1} \leq p_k$;

M2 $p_{k+1} > p_k$ and $[F_k]_{j(k)} - f_{\mathcal{N}_{k+1}}(x_{k+1}) \geq \theta_k$, where $p_{k+1} = \pi_{j(k)} \in \Pi$;

M3 $p_{k+1} > p_k$ and p_{k+1} has not been used before.

Now, let $\bar{p} = \pi_{\bar{j}} \in \Pi$ be the maximal probability that is used at infinitely many iterations. Furthermore, define the set of iterations K_1 at which the sample size changes to $\bar{p}$. The definition of $\bar{p}$ implies that there exists n_6 such that for every $k \in K_1$, $k \geq n_6$ the probability is increased to $\bar{p}$, i.e.

$$p_k < p_{k+1} = \bar{p} = \pi_{\bar{j}},\ k \geq n_6.$$

Define $r = \max\{n_5, n_6\}$ and set $K_2 = K_1 \bigcap \{r, r+1, \ldots\}$. Clearly, each iteration in K_2 excludes the case M1. Moreover, taking out the first member of a sequence K_2 and retaining the same notation for the remaining sequence we can exclude the case M3 as well. This leaves us with M2 as the only possible scenario for iterations in K_2. Therefore, for every $k \in K_2$ the following is true

$$[F_k]_{\bar{j}} - f_{\mathcal{N}_{k+1}}(x_{k+1}) \geq \theta_k \geq \theta.$$

The number of subsets of the whole set of nodes is finite. Therefore, there exists a subset $\bar{\mathcal{N}}$ which appears infinitely many times within K_2. So, define $K_3 \subseteq K_2$ such that for every $k \in K_3$

$$\mathcal{N}_{k+1} = \bar{\mathcal{N}}.$$

Denote $l_k = l(k)$ and $K_3 = \{l_k - 1\}_{k \in \mathbb{N}}$.

Then for arbitrary $k \in \mathbb{N}$ we have

$$[F_{l(k)-1}]_{\bar{j}} - f_{\bar{\mathcal{N}}}(x_{l(k)}) \geq \theta.$$

Consider $f_{\bar{\mathcal{N}}}(x_{l(k-1)})$. According to the definition of K_3 we know that $f_{\bar{\mathcal{N}}}(x_{l(k-1)})$ is used in step S2 of Algorithm 1 for updating $[F_{l(k)-1}]_{\bar{j}}$. Therefore,

$$f_{\bar{\mathcal{N}}}(x_{l(k-1)}) \geq [F_{l(k)-1}]_{\bar{j}}$$

and, for every $k \in \mathbb{N}$

$$f_{\bar{\mathcal{N}}}(x_{l(k-1)}) - f_{\bar{\mathcal{N}}}(x_{l(k)}) \geq \theta.$$

However, this implies that $f_{\bar{\mathcal{N}}}$ is decreased for a positive constant infinitely many times, which is in contradiction with the assumption A1. We conclude that the statement holds with $\bar{k} = n_4$. ■

Remark. It is important to notice here that in Theorem 3.1 $\bar{k} = \bar{k}(\omega)$ is a random variable since it depends on the outcome ω. Therefore, the value that $\bar{k}$ takes depends on a particular sample realization of the random sequence $\mathcal{N}_k$ and may be different in each particular run of the optimization procedure. We have just proved that it is finite for any given run. Notice that this statement does not imply that $\bar{k}$ is uniformly bounded across all the trajectories. Given the complexity of randomness, it might be very hard (or even impossible) to verify uniform boundedness under the current assumptions.

Under the conditions stated in Theorem 3.1, one can easily prove the following statement.

Theorem 3.2. *Let A1 and C1-C3 hold and assume that $\{x_k(\omega)\}$ is the sequence of random vectors generated with Algorithm 3. Then, for each fixed outcome ω, the sequence $\{x_k(\omega)\}$ is either finite with $n_1(\omega)$ elements and $x_{n_1}(\omega)$ is the stationary point of the objective function f, or $\{x_k(\omega)\}$ is infinite and there exists a finite $\bar{k}(\omega) \in \mathbb{N}$ such that $p_k(\omega) = 1$ for every $k \geq \bar{k}(\omega)$ and the sequence $\{x_k(\omega)\}_{k \geq \bar{k}(\omega)}$ belongs to the level set*

$$\mathcal{L}(\omega) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_{\bar{k}}(\omega)) + \tilde{e}\},$$

where $\tilde{e}$ is given in (3).

In order to prove the main theorem, we need the following assumption on the search directions. Again, we assume that this holds for an arbitrary outcome ω.

A 2. *There exists $c > 0$, such that, for arbitrary outcome ω, there holds: $d_k^T \nabla f_{\mathcal{N}_k}(x_k) \leq -c\|\nabla f_{\mathcal{N}_k}(x_k)\|^2$ for every k.*

Notice that here we assume that c is a deterministic constant, i.e. independent of ω. The inequality in assumption A2 is satisfied for the choice $d_k = -\nabla f_{\mathcal{N}_k}(x_k)$. It is also satisfied for a Quasi-Newton direction which retains uniformly positive definite inverse Hessian approximation. The sequence of search directions is bounded, e.g., if the level set $\mathcal{L}$ is compact (see Theorem 3.2); this is true if, e.g., f is convex and coercive ($f(x) \to +\infty$ whenever $\|x\| \to +\infty$).

We impose an additional standard assumption on search directions.

A 3. *For every outcome ω, the following implication is true: the sequence of search directions $\{d_k\}_{k \in \mathbb{N}}$ is bounded if the sequence $\{x_k\}_{k \in \mathbb{N}}$ is bounded.*

Notice that the above Assumption, for any ω, holds if the level set $\mathcal{L}(\omega)$ is compact. The Assumptions A2–A3 allow us to prove the main result stated below which corresponds to the classical result from deterministic optimization. We omit the arguments which relay on standard analysis of line search methods. For more details, see [30] for example.

Theorem 3.3. *Let A1-A3 and C1-C3 hold. Then, the sequence* $\{x_k(\omega)\}$ *generated by Algorithm 3 is either unbounded, or every accumulation point of the sequence is stationary for function* f.

Proof. Let us fix an arbitrary outcome ω. We have already showed that the main algorithm terminates at x_k only if $\nabla f(x_k) = 0$. Therefore, we consider the case where the number of iterations is infinite. In that case, Theorem 3.1 implies the existence of iteration $\bar{k}$ such that $p_k = 1$ for every $k \geq \bar{k}$. This means that $\tilde{N}_k = N$ for every $k \geq \bar{k}$ and furthermore implies that $f_{\mathcal{N}_k}(x) = f_{\mathcal{N}}(x) = f(x)$ for every k large enough. Therefore, the rest of the proof follows the steps of the standard nonmonotone line search analysis applied on the original objective function f and the result follows by the standard arguments. ■

Notice that Theorem 3.3 implies that Algorithm 3 converges to a stationary point of f for every ω, if f is convex and coercive.

3.2 Non-asymptotic analysis

Within this subsection we investigate conditions for the linear convergence and provide a worst-case complexity analysis under the following assumption.

A 4. *There exist positive constants* μ *and* L *such that the following holds for every* $i = 1, ..., N$ *and every* $x \in \mathbb{R}^n$

$$\mu I \preceq \nabla^2 f_i(x) \preceq LI.$$

We set $d_k = -\nabla f_{\mathcal{N}_k}(x_k)$ for simplicity, but the same analysis can be conducted for any other search direction satisfying assumption A2 and $\|d_k\| \leq R\|\nabla f_{\mathcal{N}_k}(x_k)\|$ for some $R > 0$.

The strong convexity assumption implies that the following holds for every $x \in \mathbb{R}^n$

$$\mu(f(x) - f(x^*)) \leq \|\nabla f(x)\|^2, \tag{10}$$

where x^* is an unique minimizer of f. Moreover, it also implies that the step size α_k is uniformly bounded from bellow (see [31] for instance), i.e.,

$$\alpha_k \geq \min\{1, (1-\eta)/L\} := a. \tag{11}$$

Considering (conditional) Binomial distribution of $|\mathcal{N}_k|$, one can show that for all k and $j = k, k+1$

$$E(f_{\mathcal{N}_k}(x_j)) = E(r_k f(x_j)) \quad \text{and} \quad E(\nabla f_{\mathcal{N}_k}(x_k)) = E(r_k \nabla f(x_k)), \tag{12}$$

where the expectation is taken with respect to outcome ω and $r_k = 1 - (1-p_k)^N := 1 - h_k$. Notice that $h_k \leq (1 - p_0^{min})^N := h$ and let us denote $r := 1 - h$. Then we can show the following.

Theorem 3.4. *Let A1, A4 and C1-C3 hold. Then*

$$E(f(x_{k+1}) - f(x^*)) \leq \rho E(f(x_k) - f(x^*)) + e_k/r,$$

where $\rho \in (0,1)$ provided that h is small enough.

Proof. Applying the expectation to (2) we obtain

$$E(r_k f(x_{k+1})) \leq E(r_k f(x_k)) - \eta a E(\|\nabla f_{\mathcal{N}_k}(x_k)\|^2) + e_k. \tag{13}$$

If we denote the history of Algorithm 3 until the kth iteration by $\mathcal{F}_k$, we obtain

$$E(\nabla f_{\mathcal{N}_k}(x_k)|\mathcal{F}_k) = r_k \nabla f(x_k)$$

which yields

$$r_k^2 \|\nabla f(x_k)\|^2 = \|E(\nabla f_{\mathcal{N}_k}(x_k)|\mathcal{F}_k)\|^2 \leq (E(\|\nabla f_{\mathcal{N}_k}(x_k)\||\mathcal{F}_k))^2 \leq E(\|\nabla f_{\mathcal{N}_k}(x_k)\|^2|\mathcal{F}_k).$$

Taking the expectation we get

$$-E(\|\nabla f_{\mathcal{N}_k}(x_k)\|^2) \leq -E(r_k^2 \|\nabla f(x_k)\|^2).$$

Putting the previous inequality into (13), subtracting $f(x^*)$ from both sides and using the strong convexity, we obtain

$$\begin{aligned} E(f(x_{k+1}) - f(x^*)) - E(h_k f(x_{k+1})) \quad &\leq \quad E(f(x_k) - f(x^*)) - E(h_k f(x_k)) \\ &+ \quad e_k - \eta a E(\mu(f(x_k) - f(x^*)) r_k^2). \end{aligned}$$

Rearranging the previous expression yields

$$\begin{aligned} E(f(x_{k+1}) - f(x^*)) &\leq E((f(x_k) - f(x^*))(1 - \eta a \mu r_k^2)) + e_k \\ &+ E(h_k(f(x_{k+1}) - f(x_k) \pm f(x^*))) \\ &= E((f(x_k) - f(x^*))(1 - \eta a \mu r_k^2 - h_k)) + e_k \\ &+ E(h_k(f(x_{k+1}) - f(x^*))) \\ &\leq E((f(x_k) - f(x^*))(1 - \eta a \mu r^2)) + e_k \\ &+ E(h(f(x_{k+1}) - f(x^*))). \end{aligned} \tag{14}$$

Now, using the fact that h and r are deterministic, there follows

$$E(f(x_{k+1}) - f(x^*)) \leq E(f(x_k) - f(x^*)) \frac{1 - \eta a \mu r^2}{r} + \frac{e_k}{r},$$

so the statement holds with $\rho = (1 - \eta a \mu r^2)/r \in (0, 1)$ for h small enough,i.e., for $h \in (0, (2\eta a \mu + 1 - \sqrt{4\eta a \mu + 1})/(2\eta a \mu))$. ■

Using the above result, one can prove the following complexity result (see [5],Theorem 2.3).

Theorem 3.5. *Let A1, A4 and C1-C3 hold. Then, if $\{e_k\}$ tends to zero R-linearly, Algorithm 3 takes at most $\bar{k}$ iterations to ensure $f(x_k) - f(x^*) < \varepsilon$ where*

$$\bar{k} = \lceil \frac{\log(f(x_0) - f(x^*) + Q)}{|\log(\hat{\rho})|} \log(\varepsilon^{-1}) \rceil, \quad Q > 0, \ \hat{\rho} \in (0, 1).$$

Although this result does not require achieving full sample size to reach nearly-optimal solution, the sample size lower bound represented by h can be very large. On the other hand, the proposed algorithm achieves the full sample after a finitely many iterations and, depending on the search direction and the line search, linear or even faster convergence rate may be achieved. However, it is very complicated to estimate the number of iterations needed for achieving the full sample and the full complexity result remains an open problem. One of the possible solutions is to control the sample size lower bound sequence in some predetermined way, but this is not coherent with the adaptive framework which is proposed.

4 Numerical results

The main motivation behind the proposed algorithm is to show that the class of problems considered in this paper can be solved with reduced communica-

tion and computational costs and still provide an approximate solution of the same quality as if all the workers are constantly active. In order to demonstrate the benefits of the proposed feedback approach – abbreviated here the VSS-F – it is compared with the Sample Average Approximation method (SAA) which uses the whole set of workers at every iteration ($N_k = N$ for every k). Moreover, in order to demonstrate that the feedback that controls the sample size in VSS-F can also provide some savings compared with the predetermined sequences of sample sizes, we also compare VSS-F with the heuristic scheme (referred here to as HEUR) tested in [22] where the (expected) sample size is increased by 10% at each iteration, i.e., $p_{k+1} = \min\{1.1p_k, 1\}$. Notice that we set the value of the augmentation parameter to 1.1, as used in [22]. This may not be the optimal value, and it may be possible to tune it better for specific problems and specific problem instances; this tuning is out of this paper's scope. Nonetheless, the comparisons we carry out are fair as, with each of the three tested methods (including the proposed VSS-F) – we use reasonable (but ad hoc) choices of the tuning parameters which are kept the same across all the tested problems and across all problem instances.

We compare the methods with respect to several criteria. The first is the number of function f_i evaluations (FEV), where every component of the gradient ∇f_i is counted as one function evaluation. More precisely, the costs of calculating $f_{\mathcal{N}_k}(x)$ and $\nabla f_{\mathcal{N}_k}(x)$ are assumed to be $|\mathcal{N}_k|$ and $n|\mathcal{N}_k|$, respectively. The total count of FEVs includes both inner and outer iterations. The assumed FEV cost model is suitable if one does not consider the structure of the cost functions but might be overestimating the cost in certain special cases where a gradient evaluation may be much cheaper. For example, in binary classification problems with logistic loss functions, a gradient is obtained without significant additional computation once the function itself is evaluated.

Besides FEVs, we introduce the following comparison of the methods, accounting for the parallel architecture of the underlying computational system; see, e.g., [53, 45] for similar comparison metrics. Such system can be, e.g., a computer cluster, or a wireless sensor network (set of workers) with a fusion center (master). We associate with each inner and outer iteration of a tested algorithm (VSS-F, SAA, and HEUR) a cost $\mathcal{C}$. Namely, the cost of an iteration $\mathcal{C}$ is given by:

$$\mathcal{C} = \mathcal{C}_{\text{comp}} + r\,\mathcal{C}_{\text{comm}}, \tag{15}$$

where $\mathcal{C}_{\text{comp}}$ is the computational cost, and $\mathcal{C}_{\text{comm}}$ is the communication cost.

For a computer cluster system, it is natural that $\mathcal{C}$ corresponds to the execution time of the algorithm. On the other hand, in a wireless sensor network with battery-operated devices (where power is the most expensive resource), it is natural that $\mathcal{C}$ corresponds to the total consumed power. Due to the parallel system architecture, $\mathcal{C}_{\text{comp}}$ (either per an outer or per an inner iteration) is modeled to be the same irrespective of the number of active workers. However, $\mathcal{C}_{\text{comm}}$ depends on the number of active workers. More precisely, C_{comp} equals the total number of FEVs *per single worker* within an (either inner or outer) iteration (where FEVs are counted as described above). Further, C_{comm} equals the total number of scalars communicated *from each worker to the master and from the master to all workers* within an (inner or outer) iteration. Here, a broadcast scalar communication from the master to all workers, i.e., communication when the master sends equal messages to all workers (as for example in Step S4 of Algorithm 3), is counted as a single (n-scalars) communication. Notice that, as only active workers perform communications, C_{comm} at each inner or outer iteration is an increasing function of the number of active workers. That is, C_{comm} increases with the current effective degree (number of neighbors–active workers) of the master at a given iteration; see, e.g., [53] for a related model. We will compare the three methods (VSS-F, HEUR, and SAA) by examining their total costs until stopping, where we count the overall incurred cost across all inner and outer iterations until stopping.

The parameter r is the ratio of the cost of a single scalar transmission and a single FEV. Actual value of r depends on the problem of interest and on the underlying computational system. For a computer cluster, it can be a relatively small number, for example on the order 0.01; see, e.g., [53], for a related model. On the other hand, for a wireless sensor network, r is usually a large number; see, e.g., [45]. We also include the idealized scenarios $r = 0$ (ideal computer cluster with zero-delay communications) and $r \to \infty$ (an idealization of a wireless sensor network).[2]

All of the above methods are applied on two types of convex problems – convex quadratic and logistic losses. The algorithms are tested both on

[2]Strictly speaking, for the wireless sensor network scenario where $\mathcal{C}$ is the consumed power, a more accurate model is to include in $\mathcal{C}_{\text{comp}}$ the computational cost across all workers and the master, i.e., here we could sum the FEVs across all workers and the master for each iteration. However, for wireless sensor network, we consider the idealized scenario when $r \to \infty$, so that the computational cost is negligible and the difference between the adopted model and the one described here is irrelevant.

synthetic and real data sets. Moreover, they are also tested on non-convex quadratic losses. A more detailed description is provided below.

Previous testings of similar variable sample size schemes [31] suggest that the proposed line search rule fits well with the BFGS search direction. Moreover, including the second order information enhances the performance of each of the three sample size schemes simulated here. Therefore, we set $d_k = -H_k \nabla f_{\mathcal{N}_k}(x_k)$ where H_k is the inverse Hessian approximation updated by the BFGS rule with H_0 being the identity matrix and the gradient difference calculated by $y_k = \nabla f_{\mathcal{N}_{k+1}}(x_{k+1}) - \nabla f_{\mathcal{N}_k}(x_k)$. Since $s_k^T y_k$ is not guaranteed to be positive, we skip the update if $s_k^T y_k < 10^{-8}$ as common in the BFGS approach. That way we ensure positive definiteness of B_k and provide a descent direction with respect to the current approximation of the objective function so calculating y_k and α_k as proposed does not deteriorate the main concept. We have also performed test with the negative gradient search direction and got the results that are in all cases significantly inferior to the results with BFGS direction. Furthermore, the difference between three tested updates of the control parameters i.e., VSS-F, SAA and HEUR was completely consistent with the differences presented for the BFGS direction. Thus the results obtained with the negative gradient direction are not included in the paper. The nonmonotone line search rule in (2) is implemented with $e_0 = 0.1$ and $e_k = e_0 k^{-1.1}$. In order to make a fair comparison, the search direction, the line search rule, and all the other relevant parameters are common for each of the three tested schemes. The difference is only in updating the sequence of samples and their sizes, i.e., in updating p_k. We have employed the precision measure ε_k defined by (5), while the remaining safeguard parameters are set to

$$\theta_k = (k+1)p_k, \ \gamma_k = (Nexp(1/k))^{-1}.$$

Further, we use $\nu_1 = 1/\sqrt{N}, p_0^{min} = 0.1, \ \eta = 10^{-4}, \ \beta = 0.5$ in Algorithm 2 and Algorithm 3. The set of admissible values for the control parameters p_k is determined by the step $1/N$, i.e., $\pi_{j+1} = \pi_j + 1/N$. The set of parameters shown here is used across all problems and all problem instances, i.e., no parameter tuning is performed according to each specific instance.

The convex quadratic problem is of the form

$$f_i(x) = \frac{1}{2}(x - a_i)^\top B_i (x - a_i),$$

where $B_i \in \mathbb{R}^{n \times n}$ is a positive definite (symmetric matrix), and $a_i \in \mathbb{R}^n$ is a vector uniformly distributed on the interval $[1, 11]$. Quantities $B_i, a_i, \ i =$

$1, ..., N$ are generated mutually independently. For each i, we first generate a matrix $\tilde{B}_i$ with i independent, identically distributed (i.i.d.) elements that have the standard normal distribution. Then, we extract the eigenvector matrix $Q_i \in \mathbb{R}^{n\times n}$ of matrix $\frac{1}{2}(\tilde{B}_i + \tilde{B}_i^\top)$. We set $B_i = Q_i \text{Diag}(c_i) Q_i^\top$, where $c_i \in \mathbb{R}^n$ has the i.i.d. entries uniformly distributed on the interval $[1, 101]$.

The non-convex case takes the same form but with the B_i's entries generated to be i.i.d., from the normal distribution with mean 5 and variance 1. Notice that, in this case, the desired objective function $f(x)$ can be written as $f(x) = \frac{1}{2}x^T Ax - bx + c$, for some (possibly indefinite) $n \times n$ matrix A, vector $b \in \mathbb{R}^n$ and scalar c. Here, the function f may not be convex and hence we can not claim convergence towards global minimizer. However, we are interested in finding a stationary point of f. As it is well know, see [39], the iterative sequence is convergent if it is bounded and then it converges to a stationary point. The practical value of this is, e.g., in solving in a distributed manner (over the master-worker architecture) the linear system $Ax = b$ with an indefinite matrix A, which is an important problem in its own right, see [21, 49, 16]. Although this example is not covered by the theoretical results obtained in the previous Section, the corresponding numerical results indicate that the algorithm behaves rather well on this example as well.

The logistic problem is a machine learning problem of determining the linear classifier which minimizes the logistic loss. We give a brief description, while more details can be found in [9] for instance. The local loss function is

$$f_i(x) = R\|x\|^2 + \sum_{j=1}^{J} \ln\left(1 + e^{-b_i^j x^T a_i^j}\right)$$

where $a_i^j \in \mathbb{R}^n$, $b_i^j \in \mathbb{R}$ with the nth component equal to 1, i.e. $[a_i^j]_n = 1$, and R is a regularization parameter. In the case of synthetic data, the remaining components of a_i^j are generated randomly and independently by using the standard normal distribution. Moreover, we form $\bar{x} \in \mathbb{R}^n$ also with the standard normal distribution and define $b_i^j = sgn(\bar{x}^T a_i^j + \rho_{ij})$ where the ρ_{ij}'s are independent, with the Gaussian distribution having zero mean and standard deviation 0.1. We also perform testings on the real data set "mushrooms" which is available at the following repository: `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`.

All these problems are generated in advance, before the optimization process starts. Therefore, the objective function is considered as deterministic.

We conducted 10 runs of each of the three methods for all the problems (convex quadratic – synthetic data, logistic – synthetic and real data, and non-convex quadratic – synthetic data). For the synthetic data, each test is performed with a different problem instance where all the input parameters are re-generated as explained above. With the real data set "mushrooms," 10 tests are repeated with the same input data. However, the difference between the tests arises because VSS-F and HEUR yield a stochastic sequence of iterations and therefore generate different results at each test (each run). With each of the three methods (VSS-F, SAA, and HEUR), we use the initialization $x_0 = 0$. The stopping criterion, with each of the three methods, is as follows: the algorithms terminate if the maximal number of FEVs (10^7) is reached or if

$$\|\nabla f_{\mathcal{N}_k}(x_k)\| < \tau \quad \text{and} \quad \mathcal{N}_k = \mathcal{N},$$

where we use different values of τ for different problems (convex quadratic – synthetic data, logistic – synthetic and real data, and non-convex quadratic – synthetic data). Notice that, with this stopping rule, the algorithms can terminate only if the master detects that the full sample $\mathcal{N}$ is used. The full sample is required because no knowledge on the similarity of the f_i's is assumed at the master. Indeed, having $\|\nabla f_{\mathcal{N}_k}(x_k)\|$ small, with $|\mathcal{N}_k| < N$, may not imply that $\|\nabla f(x_k)\|$ is small as well. Designing stopping criteria under the imposed f_i-similarity is left for future work.

In the convex quadratic case, the stopping criterion parameter is $\tau = 0.5$ which yields the relative error in the objective function of order 10^{-6}. More precisely, $(f(x_k) - f(x^*))/f(x^*)$, where $x^* = \arg\min_x f(x)$, is of the order 10^{-6}. The dimension of the optimization variable is $n = 3$ and the number of nodes is $N = 1000$. These parameters are retained in a general (non-convex) case as well.

In the logistic - synthetic case, the stopping criterion is defined by $\tau = 10^{-2}$, the dimension is $n = 4$, the number of nodes is $N = 1000$, and the number of samples per node is $J = 2$. With the data set "mushrooms," the number of features is $n = 112$, the number of data points, here equal to the number of workers, is $N = 8124$, and $J = 1$. The regularization parameter R is set to 0.1. Since none of the tested methods managed to converge within 10^7 FEVs (except for VSS-F in one run) we set $\tau = 10^{-1}$ and report the relevant results. Given that the cost of a gradient evaluation increases with the dimension of the optimization variable n, a larger FEV limit than the adopted value of 10^7 might be considered for this problem. However, the

reported results with the 10^7 FEV limit and $\tau = 0.1$ give enough insights of the benefits obtained through the proposed feedback scheme.

The results on the analysis of FEVs are shown at Figure 2. The (green) diamonds show the costs for each run of VSS-F, the (red) dots are the costs of SAA while the (blue) squares are the costs of HEUR, for each run, expressed as the number of function evaluations, FEV. The cost reduction obtained by VSS-F with respect to both competitors is rather significant. For convex quadratic the savings are ranging form 20% to 50%, Figure 2.1. In the case of non-convex quadratic the savings are even larger, going up to 80%, Figure 2.2. The same is happening with the synthetic logistic loss example - we have the reduction of up to 50%, Figure 2.3. In 2 runs considering the real data set, none of the tested schemes converged within 10^7 FEVs. For the remaining runs, the savings are from 25% to 70% regarding SAA and similar results hold for HEUR.

It is important to comment here the role of safeguard parameters in the actual implementation of the algorithm. As already explained p_k^{min} is important from the theoretical point of view as this sequences ensures that the full set of nodes is employed eventually. Thus it might seem that p_k^{min} actually interferes with the feedback mechanism and thus even reduces the relatively complex mechanism of p_k update into mere increase by a certain schedule. Apparently, that does not happen in actual implementation. In all tests we have performed $p_k^{\min}$ did not significantly affect the feedback process at all as it value increased only three times within all runs in all examples. Thus the vast majority of runs terminated with $p_k^{\min} = p_0^{\min} = 0.1$.

We also report on the number of inner iterations that the proposed VSS-F method incurs on the four tested problems is usually quite small. Namely, on each of the four tested problems, the number of inner iterations per outer iteration is typically 3-4 and maximally 5 (across all runs of all four tested problems).

Figures 3 and 4 show comparisons with respect to cost (15) for the strongly convex quadratic example and the logistic losses example (synthetic data), respectively. The Figures for the remaining two considered examples (non-convex quadratic losses, and logistic losses for the "mushrooms" data set) are similar and are hence omitted for brevity. We consider three scenarios: 1) $r = 0$ – an idealized setting for execution time on a computer cluster; 2) $r = 0.01$ – a potential setting for execution time in a real computer cluster; and 3) $r \to \infty$ – an idealized setting for a wireless sensor network. The figures present the overall costs (counting both the costs of inner and

outer iterations) until stopping for the three values of r (Figures 3.1 and 4.1: $r = 0$; Figures 3.2 and 4.2: $r = 0.01$; Figures 3.3 and 4.3: $r \to \infty$[3]) The stopping criterion and the stopping parameters are the same as in the FEVs comparisons above.

We now comment on the obtained results. Consider first Figure 3.1. Here, as the communication cost (time) is neglected and as the time per inner or per outer iteration does not depend on the number of active workers, it is natural that SAA performs the best, as it performs the "most exact" updates among the three tested methods. However, when the communication cost is no more negligible (Figures 3.2 and 3.3), the proposed VSS-F performs better than both SAA and HEUR. Similar conclusions can be drawn from the logistics losses example as well (Figure 4.)

We end this section by illustrating the sample size behavior (Figure 5) and the progress of the algorithm in terms of the full gradient norm against the relevant cost measure compared to other tested schemes (Figure 6).

5 Conclusion

We considered a generic nonmonotone line search framework for the minimization of a finite sum of generic component costs. We assumed a master-worker model, where the master maintains the solution estimate x_k and broadcasts it to the workers. Each worker holds a component cost, and, at each iteration k, it can be in one of the two possible states – active or inactive. At each iteration k, active workers evaluate their component costs and the corresponding gradients at the current iterate x_k and send this information back to the master, while inactive workers stay idle. We proposed a mechanism in which the master node controls the average number of active workers through a single scalar parameter p_k. The master adaptively increases or decreases p_k over iterations, as needed, based on an inexpensive estimate of the algorithm progress. Hence, the master sends through p_k the feedback information to the workers as to how many of them should be active at the next iteration to ensure progress, on the one hand, and save computational cost as much as possible, on the other hand. Simulations on convex and non-convex quadratic losses and on (convex) logistic losses – both on

[3]For the case $r \to \infty$, computational cost becomes negligible; hence, we plot on the y-axis only the communication cost, for convenience.

synthetic and real world data sets – demonstrate the benefits of introducing feedback in the sample size choice along iterations.

There are several interesting future research directions. First, the framework can be specialized and analyzed to more restricted classes of costs. Second, it is certainly relevant to incorporate in the master-worker model the effects of several imperfections, like delays and network topology. For example, if the (master and worker) nodes are organized in a network (e.g., a two dimensional grid), then the gradient and function information from certain workers will be received with delays, depending on how distant (in terms of the number of hops) a worker is from the master; see, e.g., [1]. In this paper, we considered the idealized model, as it represents a necessary starting point for the analysis of the feedback mechanisms in the control of the sample size. Studying effects of various imperfections like delays represents an interesting future research direction.

Acknowledgement. We are grateful to the anonymous referees whose comments and suggestions helped us to improve the quality of this paper.

References

[1] A. Agarwal, J. Duchi, Distributed Delayed Stochastic Optimization, *Proceedings of the Advances in Neural Information Processing Systems, (2011).*

[2] F. Bastin, Trust-Region Algorithms for Nonlinear Stochastic Programming and Mixed Logit Models, *PhD thesis, University of Namur, Belgium, 2004.*

[3] F. Bastin, C. Cirillo, P. L. Toint, An adaptive Monte Carlo algorithm for computing mixed logit estimators, *Computational Management Science 3(1), (2006), pp. 55-79.*

[4] F. Bastin, C. Cirillo, P. L. Toint, Convergence theory for nonconvex stochastic programming with an application to mixed logit, *Math. Program., Ser. B 108, (2006), pp. 207-234.*

[5] S. Bellavia, N. Krejić, N. Krklec Jerinkić, Subsampled inexact new-ton methods for minimizing large sums of convex functions, *arXiv:1811.05730, (2018).*

[6] A. Berahas, J. Nocedal, M. Takac, A multi-batch L-BFGS method for machine learning, *Advances in Neural Information Processing Systems, (2016), pp. 1055–1063.*

[7] D. P. Bertsekas, Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey, *Report LIDS – 2848, (2010), pp. 85-119.*

[8] E. G. Birgin, J. M. Martínez, M. Raydan, Nonmonotone Spectral Projected Gradient Methods on Convex Sets *SIAM J. Optim. 10(4), (2006), pp. 1196-1211.*

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends in Machine Learning, 3(1), (2011), pp. 1-122.*

[10] R. H. Byrd, G. M. Chin, W. Neveitt, J. Nocedal, On the Use of Stochastic Hessian Information in Optimization Methods for Machine Learning, *SIAM J. Optim., 21(3), (2011), pp. 977-995.*

[11] R. H. Byrd, G. M. Chin, J. Nocedal, Y. Wu, Sample size selection in optimization methods for machine learning, *Mathematical Programming, 134(1), (2012), pp. 127-155.*

[12] R. Bollapragada, R. Byrd, J. Nocedal, Adaptive sampling strategies for stochastic optimization, *arXiv preprint arXiv:1710.11258, (2017).*

[13] R. Bollapragada, D. Mudigere, J. Nocedal, H.-J. Michael Shi, P. Tak Peter Tang, A progressive batching l-bfgs method for machine learning,*arXiv preprint arXiv:1802.05374, (2018).*

[14] R. Byrd, G. Chin, J. Nocedal, Y. Wu, Sample size selection in optimization methods for machine learning, *Mathematical programming, 134(1), (2012), pp. 127–155.*

[15] R. Byrd, S. Hansen, J. Nocedal, Y. Singer, A stochastic quasi-Newton method for large-scale optimization, *SIAM Journal on Optimization, 26(2), (2016), pp. 1008–1031.*

[16] S.-C. T. Choi, C. C. Paige, M. A. Saunders, MINRES-QLP: A Krylov Subspace Method for Indefinite or Singular Symmetric Systems, *SIAM Journal on Scientific Computing, 33(4),(2011), pp. 1810-1836.*

[17] F. Curtis, A self-correcting variable-metric algorithm for stochastic optimization, *In International Conference on Machine Learning, (2016), pp. 632–641.*

[18] A. Defazio, F. Bach, S. Lacoste-Julien, Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, *In Advances in neural information processing systems, (2014), pp. 1646–1654.*

[19] E. Dolan, J. More, Benchmarking optimization software with performance profiles, *Mathematical programming, 91(2), (2002), pp. 201-213.*

[20] M.A. Diniz-Ehrhardt, J. M. Martínez, M. Raydan, A derivative-free nonmonotone line-search technique for unconstrained optimization, *Journal of Computational and Applied Mathematics, 219(2), (2008), pp. 383-397.*

[21] R. Fletcher, Conjugate Gradient Methods for Indefinite Systems, *Numerical Analysis, Springer, (1976), pp. 73-89.*

[22] M. P. Friedlander, M. Schmidt, Hybrid deterministic-stochastic methods for data fitting, *SIAM J. Scientific Computing, 34(3), (2012), pp. 1380-1405.*

[23] D. H. Li, M. Fukushima, A derivative-free line search and global convergence of Broyden-like method for nonlinear equations, *Opt. Methods Software, 13, (2000), pp. 181-201.*

[24] R. Gower, D. Goldfarb, P. Richtarik, Stochastic block BFGS: Squeezing more curvature out of data, *International Conference on Machine Learning, (2016), pp. 1869–1878.*

[25] M. Gurbuzbalaban, A. Ozdaglar, P. Parrilo, A globally convergent incremental Newton method, *Mathematical Programming, 151(1), (2015), pp. 283–313.*

[26] Ghadimi, S., Lan, G., Zhang, H, Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization, *Mathematical Programming, 155(1-2), (2016), pp. 267–305.*

[27] L. Grippo, F. Lampariello, S. Lucidi, A nononotone line search technique for Newton's method, *SIAM J. Numerical Analysis, 23(4), (1986), pp. 707-716.*

[28] F.S. Hashemi, S. Ghosh, R. Pasupathy, On adaptive sampling rules for stochastic recursion, *Proceedings of the Winter Simulation Conference 2014, IEEE, (2014), pp. 3959-3970.*

[29] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, *Advances in neural information processing systems, (2013), pp. 315–323.*

[30] N. Krejić, N. Krklec, Line search methods with variable sample size for unconstrained optimization, *Journal of Computational and Applied Mathematics, 245, (2013), pp. 213-231.*

[31] N. Krejić, N. Krklec Jerinkić, Nonmonotone line search methods with variable sample size, *Numerical Algorithms, 68(4), (2015), pp. 711-739.*

[32] N. Krejić, N. Krklec Jerinkić, Stochastic gradient methods for unconstrained optimization, *Pesquisa Operacional, 34(3), (2014), pp. 373-393.*

[33] J. Langford, A. Smola, M. Zinkevich, Slow learners are fast, *Proc. Advances in Neural Information Processing Systems 22, (2009), pp. 2331-2339.*

[34] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, B.-Y. Su, Scaling Distributed Machine Learning with the Parameter Server *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), (2014), pp. 583-598.*

[35] A. Mokhtari, M. Eisen, A. Ribeiro, IQN: An incremental quasi-Newton method with local superlinear convergence rate, *SIAM Journal on Optimization, 28(2), (2018), pp. 1670–1698.*

[36] A. Mokhtari, A. Ribeiro, Global convergence of online limited memory BFGS, *Journal of Machine Learning Research, 16(1), (2015), pp. 3151–3181.*

[37] A. Nedic, D. P. Bertsekas, Incremental Subgradient Methods for Nondifferentiable Optimization, *SIAM J. Optimization, 12, (2001), pp. 109-138.*

[38] A. Nedic, A. Olshevsky, Stochastic Gradient-push for Strongly Convex Functions on Time-varying Directed Graphs, *IEEE Transactions on Automatic Control, 61(12), (2016), pp. 3936-3947.*

[39] J. Nocedal, S. J. Wright, Numerical Optimization, *Springer, 1999.*

[40] L. Nguyen, J. Liu, K. Scheinberg, M. Takac, SARAH: A novel method for machine learning problems using stochastic recursive gradient, *International Conference on Machine Learning, (2017), pp. 2613–2621.*

[41] R. Pasupathy, On choosing parameters in retrospective-approximation algorithms for simulation-optimization, *Proceedings of the 2006 Winter Simulation Conference, IEEE, (2006), pp. 208-215.*

[42] R. Pasupathy, On Choosing Parameters in Retrospective-Approximation Algorithms for Stochastic Root Finding and Simulation Optimization, *Operations Research, 58(4), (2010), pp. 889-901.*

[43] C. Paquette, K. Scheinberg, A stochastic line search method with convergence rate analysis, *arXiv preprint arXiv:1807.07994, (2018).*

[44] E. Polak, J. O. Royset, Efficient sample sizes in stochastic nonlinear programing, *Journal of Computational and Applied Mathematics, 217(2), (2008), pp. 301-310.*

[45] M. Rabbat, R. Nowak, Distributed Optimization in Sensor Networks, *Proceedings of the 3rd international symposium on Information processing in sensor networks (2004).*

[46] B. Recht, C. Re, S. J. Wright, F. Niu, Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent, *Advances in neural information processing systems, (2011), pp. 693-701.*

[47] J. O. Royset, Optimality functions in stochastic programming, *Math. Programming, 135(1-2), (2012), pp. 293–321.*

[48] J. Royset, R. Szechtman, Optimal budget allocation for sample average approximation, *Operations Research, 61(3), (2013), pp. 762–776.*

[49] Y. Saad, Iterative Solution of Indefinite Symmetric Linear Systems by Methods Using Orthogonal Polynomials over Two Disjoint Intervals, *SIAM J. Numer. Anal., 20(4), (1983), pp. 784–811.*

[50] M. Schmidt, N. Le Roux, F. Bach, Minimizing Finite Sums with the Stochastic Average Gradient, *Mathematical Programming, 162(1-2), (2017), pp. 83–112.*

[51] O. Shamir, N. Srebro, T. Zhang, Communication Efficient Distributed Optimization using an Approximate Newton-type Method, *31st International Conference on Machine Learning, ICML (2014).*

[52] N. Schraudolph, J. Yu, S. Gunter, A stochastic quasi-Newton method for online convex optimization, *Artificial Intelligence and Statistics, (2007), pp. 436–443.*

[53] K. I. Tsianos, S. Lawlor, M. G. Rabbat, Communication/Computation Tradeoffs in Consensus-Based Distributed Optimization, *Advances in neural information processing systems, (2012), pp. 1943-1951.*

[54] Uday V. Shanbhag, J.H. Blanchet, Budget constrained stochastic optimization, *Proceedings of the 2015 Winter Simulation Conference, Huntington Beach, CA, USA, December 6-9, IEEE/ACM, (2015), pp. 368-379.*

[55] Uday V. Shanbhag, Decomposition and Sampling Methods for Stochastic Equilibrium Problems, *PhD thesis, Department of Management Science and Engineering (Operations Research), Stanford University, 2006.*

[56] F. Yousefian, A. Nedic, U. V. Shanbhag On stochastic gradient and subgradient methods with adaptive steplength sequences, *Automatica 48(1), (2012), pp. 56-67.*

[57] Y. Zhang, J. C. Duchi, M. Wainwright, Comunication-Efficient Algorithms for Statistical Optimization, *Journal of Machine Learning Research, 14(Nov), (2013), pp. 3321-3363.*

[58] H. Zhang, W. W. Hager, A nonmonotone line search technique and its application to unconstrained optimization *SIAM J. Optim. 4, (2004), pp. 1043-1056.*

[59] M. Zinkevich, M. Weimer, A. J. Smola, L. Li, Parallelized Stochastic Gradient Descent, *Proceedings of the Advances in Neural Information Processing Systems, (2010).*

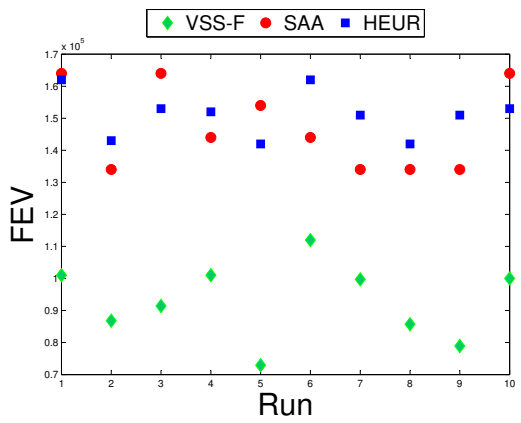

Figure 2.1: Convex quadratic

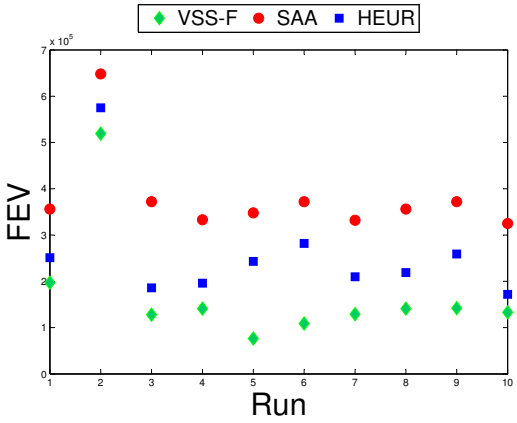

Figure 2.2: Non - convex quadratic

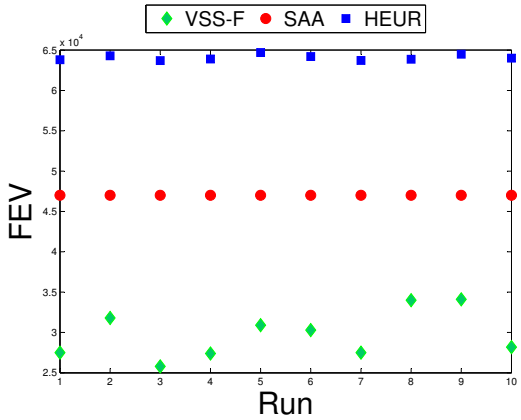

Figure 2.3: Logistic - synthetic

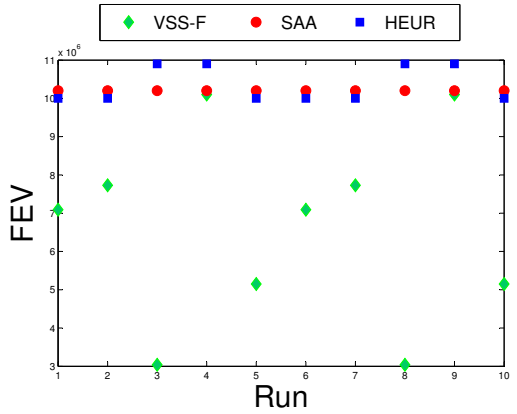

Figure 2.4: Logistic - mushrooms

Figure 2: Total number of FEVs until stopping for 10 independent runs of each tested method for the four tested problems.

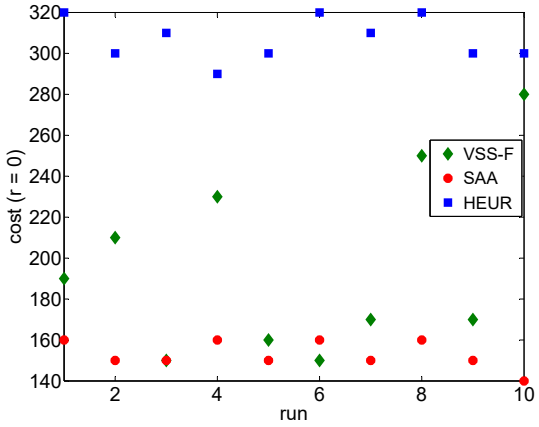

Figure 3.1

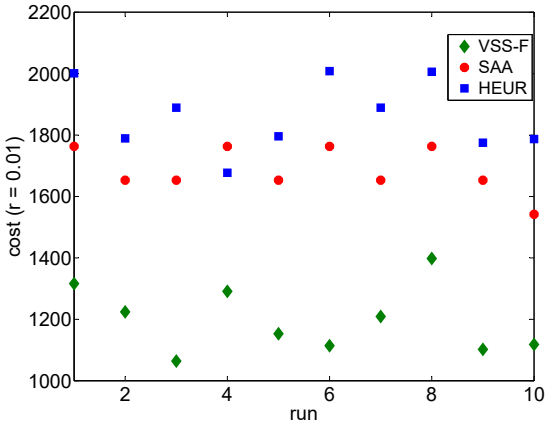

Figure 3.2

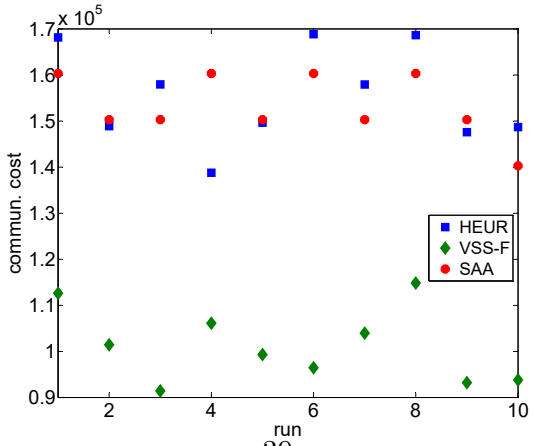

Figure 3.3

Figure 3: Total cost (15) until stopping for 10 independent runs of each tested method, for the example of strongly convex quadratic losses and different values of parameter r; Fig. 3.1: $r = 0$; Fig. 3.2: $r = 0.01$; and Fig. 3.3: $r \to \infty$ (for $r \to \infty$, the y-axis shows communication cost).

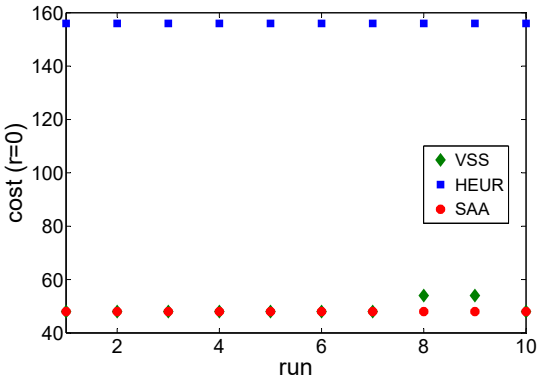

Figure 4.1

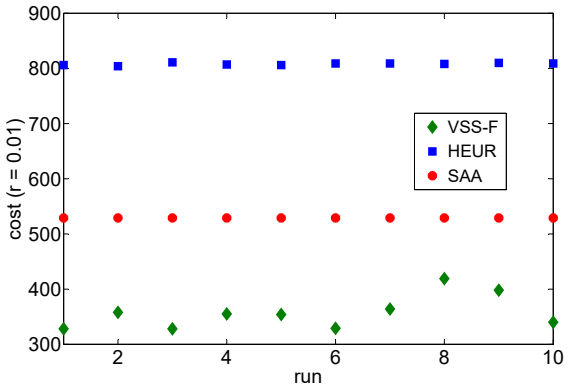

Figure 4.2

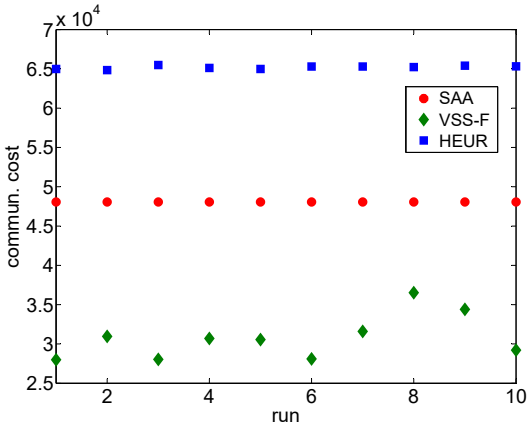

Figure 4.3

Figure 4: Total cost (15) until stopping for 10 independent runs of each tested method, for the example of logistic losses (synthetic data) and different values of parameter r; Fig. 4.1: $r = 0$; Fig. 4.2: $r = 0.01$; and Fig. 4.3: $r \to \infty$ (for $r \to \infty$, the y-axis shows communication cost).

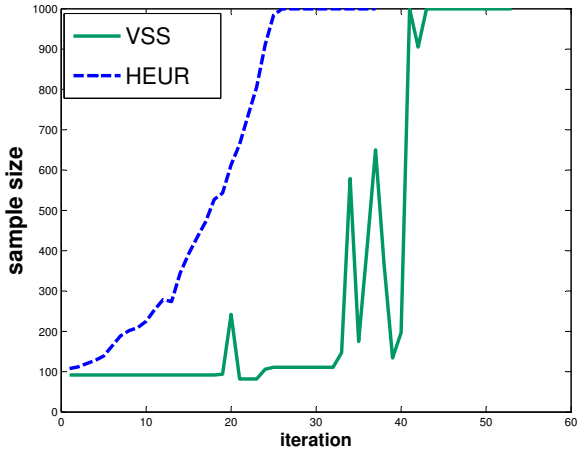

Figure 5

Figure 5: Sample size behavior of the considered algorithms for one run, the non-convex quadratic case.

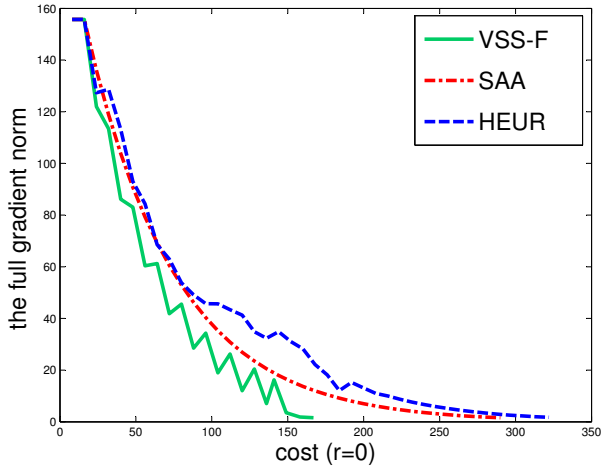

Figure 6.1

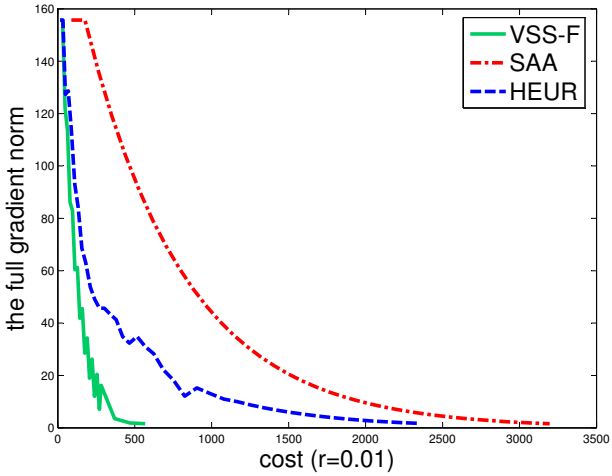

Figure 6.2

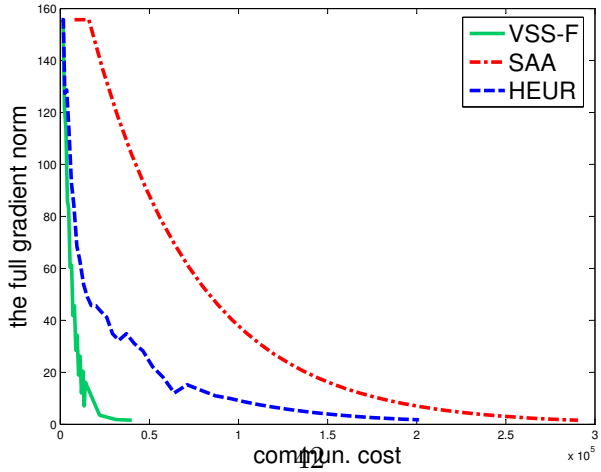

Figure 6.3

Figure 6: The full gradient norm until stopping (y-axis) for one run of each tested method, for the example of non-convex quadratic losses, against the Total cost (15) with different values of parameter r; Fig. 6.1: $r = 0$; Fig. 6.2: $r = 0.01$; and Fig. 6.3: $r \to \infty$.