Line search methods with variable sample size for unconstrained optimization

Nataša Krejić[*] Nataša Krklec[†]

June 28, 2011

Abstract

Minimization of unconstrained objective function in the form of mathematical expectation is considered. Sample Average Approximation - SAA method transforms the expectation objective function into a real-valued deterministic function using large sample and thus deals with deterministic function minimization. The main drawback of this approach is its cost. A large sample of the random variable that defines the expectation must be taken in order to get a reasonably good approximation and thus the sample average approximation method assumes very large number of function evaluations. We will present a line search strategy that uses variable sample size and thus makes the process significantly cheaper. Two measures of progress - lack of precision and decrease of function value are calculated at each iteration. Based on these two measures a new sample size is determined. The rule we will present allows us to increase or decrease the sample size in each iteration until we reach some neighborhood of the solution. An additional safeguard check is performed to avoid unproductive sample decrease. Eventually the maximal sample size is reached so the variable sample size strategy generates a solution of the same quality as

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

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

SAA method but with significantly smaller number of function evaluations. The algorithm is tested on a couple of examples including the discrete choice problem.

Key words: stochastic optimization, line search, simulations, sample average approximation, variable sample size

1 Introduction

The problem under consideration is

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1}$$

Function $f : \mathbb{R}^n \to \mathbb{R}$ is in the form of mathematical expectation

$$f(x) = E(F(x, \xi)),$$

where $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, ξ is a random vector $\xi : \Omega \to \mathbb{R}^m$ and $(\Omega, \mathcal{F}, P)$ is a probability space. The form of mathematical expectation makes this problem difficult to solve as very often one can not find its analytical form. This is the case even if the analytical form of F is known, what will be assumed in this paper.

One way of dealing with this kind of problem is to use sample average in order to approximate the original objective function as follows

$$f(x) \approx \hat{f}_N(x) = \frac{1}{N} \sum_{i=1}^{N} F(x, \xi_i). \tag{2}$$

This is the approach that we use as well. Here N represents the size of sample that is used to make approximation (2). Important assumption is that we form the sample by random vectors $\xi_1, \ldots, \xi_N$ that are independent and identically distributed. If F is bounded then Law of large numbers [18] implies that for every x almost surely

$$\lim_{N \to \infty} \hat{f}_N(x) = f(x). \tag{3}$$

In practical applications one can not have unbounded sample size but can get close to the original function by choosing a sample size that is large enough but still finite. So, we will focus on finding an optimal solution of

$$\min_{x \in \mathbb{R}^n} \hat{f}_N(x), \tag{4}$$

where N is a fixed integer and $\xi_1, \ldots, \xi_N$ is a sample realization that is generated at the beginning of optimization process. Thus the problem we are considering is in fact deterministic and standard optimization tools are applicable. This approach is called the sample path method or the stochastic average approximation (SAA) method and it is subject of many research efforts, see for example [18] and [19]. The main disadvantage of SAA method is the need to solve an individual optimization problem for each iteration with the objective function defined by (2). As N in (4) needs to be large the evaluations of $\hat{f}_N$ become very costly. That is particularly true in practical applications where the output parameters of models are expensive to calculate. Given that almost all optimization methods include some kind of gradient information or even second order information, the cost becomes even higher.

Various attempts to reduce the costs of SAA methods are presented in the literature. Roughly speaking the main idea is to use some kind of variable sample size strategy and work with smaller samples whenever possible, at least at the beginning of optimization process. One can distinguish two types of variable sample size results. The first type deals with unbounded samples and seeks convergence in stochastic sense. The strategies of this type in general start with small samples and increase their size during iterative procedure. Up to our best knowledge no such method allows us to decrease the sample size during the process. One efficient method of this kind is presented in [6]. The proposed method uses Bayesian scheme to determine a suitable sample size in each iteration within the trust region framework. It yields almost sure convergence towards a solution of (1). In general the sample size in this method is unbounded but in some special cases it can even stay bounded.

The dynamic of increasing the sample size is the main issue of papers [10] and [16] as well. In [10], convergence is ensured if (3) is satisfied and sample size posses sufficient growth. For example, the sample size that is at least as big as the square root of current iteration's number is a suitable one. On the other hand, the method includes a signal that indicates whether the increase is really necessary. For that purpose the paired t-test is being used to compare the neighboring iterations objective function values. The method in [16] states an auxiliary problem that is solved before the optimization process is started. The solution of that auxiliary problem provides an efficient variable sample size strategy. However, this strategy does not allow decrease in the sample size.

The second type of algorithms deals directly with problems of type (4) and seeks convergence towards stationary points of that problem. The algorithms proposed in [2] and [3] introduce variable sample size strategy that allows decrease of the sample size as well as increase during optimization process. Roughly speaking, the main idea is to use the decrease of function value and a measure of the width of confidence interval to determine the change in sample size. The optimization process is conducted in the trust region framework. We will adopt these ideas to the line search framework in this paper and propose an algorithm that allows both increase and decrease of sample size during the optimization process. Given that the final goal is to make the overall process less costly we also introduce an additional safeguard rule that prohibits unproductive sample decreases. As common for this kind of problems by cost we will always assume the number of function evaluations [14].

The paper is organized as follows. In Section 2 we will state the problem in more details and present the assumptions needed for the proposed algorithm. The algorithm is presented in Section 3 and convergence results are derived in Section 4. Numerical results are presented in the last section.

2 Preliminaries

In order to solve (4) we will assume that we know the analytical form of a gradient $\nabla_x F(x, \xi)$. This implies that we are able to calculate the true gradient of a function $\hat{f}_N$, that is

$$\nabla \hat{f}_N(x) = \frac{1}{N} \sum_{i=1}^{N} \nabla_x F(x, \xi_i).$$

Once the sample is generated, we observe the function $\hat{f}_N$ and the problem (4) as a deterministic one [9]. This approach simplifies the definition of stationary points which is much more complicated in stochastic environment. It also provides us with standard optimization tools. Various optimization algorithms are described in [15], for example. The one that we will apply belongs to the line search type of algorithms. The main idea is to determine a suitable direction and search along that direction in order to find a step that provides a sufficient decrease of the objective function value.

In order to make problem (4) close enough to the original problem (1), the sample size N has to be substantially large. On the other hand, in practical implementations, evaluating the function $F(x,\xi)$ can be very costly. Therefore, working with a large sample size N during the whole optimization process can be very expensive. In order to overcome this difficulty, algorithms that vary the sample size have been developed. In [3] and [2], methods that allow the sample size to decrease are proposed. In these algorithms, the trust region approach is used as a tool for finding the upcoming iteration. They are constructed to solve problem (4) with N being some finite number defined at the beginning of a process. We will follow this approach in the framework of line search methods thus allowing the sample size to vary across iterations, being increased or decreased.

Suppose that we are at the iteration x_k. Every iteration has its own sample size N_k, therefore we are observing the function

$$\hat{f}_{N_k}(x) = \frac{1}{N_k} \sum_{i=1}^{N_k} F(x, \xi_i)$$

We perform line search along the direction p_k which is decreasing for the observed function, i.e. it satisfies the condition

$$p_k^T \nabla \hat{f}_{N_k}(x_k) < 0. \qquad (5)$$

In order to obtain a sufficient decrease of the objective function, we use the backtracking technique to find a step size α_k which satisfies the Armijo condition

$$\hat{f}_{N_k}(x_k + \alpha_k p_k) \leq \hat{f}_{N_k}(x_k) + \eta \alpha_k p_k^T \nabla \hat{f}_{N_k}(x_k), \qquad (6)$$

for some $\eta \in (0,1)$. More precisely, starting from $\alpha = 1$, we decrease α by multiplying it with factor $\beta \in (0,1)$ until the Armijo condition (6) is satisfied. This can be done in a finite number of trials if the iteration x_k is not a stationary point of $\hat{f}_{N_k}$ as this function is continuously differentiable and bounded from below. For more information about this technique see [15], for example.

After the suitable step size α_k is found, we define the next iteration as $x_{k+1} = x_k + \alpha_k p_k$. Now, the main issue is how to determine a suitable sample size N_{k+1} for the following iteration. In the algorithm that we propose the rule for determining N_{k+1} is based on the decrease measure dm_k, the lack of precision denoted by $\varepsilon_\delta^{N_k}(x_k)$ and the safeguard rule parameter ρ_k. The two measures of progress, dm_k and $\varepsilon_\delta^{N_k}(x_k)$ are taken from [3] and [2] and adopted to suit the line search methods while the third parameter is introduced to avoid unproductive decrease of the sample size as will be explained below.

The decrease measure is defined as

$$dm_k = -\alpha_k p_k^T \nabla \hat{f}_{N_k}(x_k). \tag{7}$$

This is exactly the decrease in the linear model function, i.e.

$$dm_k = m_k^{N_k}(x_k) - m_k^{N_k}(x_{k+1}),$$

where

$$m_k^{N_k}(x_k + s) = \hat{f}_{N_k}(x_k) + s^T \nabla \hat{f}_{N_k}(x_k).$$

The lack of precision represents an approximate measure of the width of confidence interval for the original objective function f, i.e.

$$\varepsilon_\delta^{N_k}(x_k) \approx c,$$

where

$$P(f(x_k) \in [\hat{f}_{N_k}(x_k) - c, \hat{f}_{N_k}(x_k) + c]) \approx \delta.$$

The confidence level δ is usually equal to 0.9, 0.95 or 0.99. It will be an input parameter of our algorithm. We know that $c = \sigma(x_k)\alpha_\delta/\sqrt{N_k}$, where $\sigma(x_k)$ is a standard deviation of a random variable $F(x_k, \xi)$ and α_δ is the quantile of Normal distribution, i.e. $P(-\alpha_\delta \leq X \leq \alpha_\delta) = \delta$, where $X : \mathcal{N}(0,1)$. Usually we can not find $\sigma(x_k)$ so we use the centered sample variance estimator

$$\hat{\sigma}_{N_k}^2(x_k) = \frac{1}{N_k - 1}\sum_{i=1}^{N_k}(F(x_k, \xi_i) - \hat{f}_{N_k}(x_k))^2.$$

Finally, we define the lack of precision as

$$\varepsilon_\delta^{N_k}(x_k) = \hat{\sigma}_{N_k}(x_k)\frac{\alpha_\delta}{\sqrt{N_k}}. \tag{8}$$

The algorithm that provides us with a candidate N_k^+ for the next sample size will be described in more details in the following section. The main idea is to compare the previously defined lack of precision and the decrease measure. Roughly speaking if the decrease in function's value is large compared to the width of the confidence interval then we decrease the sample size in the next iteration. In the opposite case, when the decrease is relatively small in comparison with the precision then we increase the sample size. Furthermore, if the candidate sample size is lower than current one, that is if $N_k^+ < N_k$, one more test is applied before making the final decision about the sample size to be used in the next iteration. In that case, we calculate the safeguard parameter ρ_k. It is defined as ratio between the decrease in the candidate function and the function that has been used to obtain the next iteration, that is

$$\rho_k = \frac{\hat{f}_{N_k^+}(x_k) - \hat{f}_{N_k^+}(x_{k+1})}{\hat{f}_{N_k}(x_k) - \hat{f}_{N_k}(x_{k+1})}. \tag{9}$$

The role of ρ_k is to prevent unproductive sample size decrease i.e. we calculate the progress made by the new point and the candidate sample size and compare it with the progress achieved with N_k. So if ρ_k is relatively small we will not allow the decrease of the sample size.

Now, we present the assumptions needed for proving the convergence result of the algorithm that will be presented in Section 3 and analyzed in Section 4.

A1 Random vectors $\xi_1, \ldots, \xi_N$ are independent and identically distributed.

A2 For every ξ, $F(\cdot, \xi) \in C^1(\mathbb{R}^n)$.

A3 There exists a constant $M_1 > 0$ such that for every ξ, x

$$\|\nabla_x F(x, \xi)\| \leq M_1.$$

A4 There are finite constants M_F and M_{FF} such that for every ξ, x,

$$M_F \leq F(x, \xi) \leq M_{FF}.$$

The role of the first assumption is already clear. It ensures that our approximation function $\hat{f}_{N_k}$ is, in fact, a centered estimator of the function f at each point. This is not a fundamental assumption that makes the upcoming algorithm convergent, but it is important for making the problem (4) close to the original one for N large enough.

The assumption A2 ensures the continuity and differentiability of F as well as of $\hat{f}_N$.

One of the crucial assumptions for proving the convergence result is A4. Moreover, A4 makes our problem solvable.

An important consequence of the previous assumptions is that the interchange between the mathematical expectation and the gradient operator is allowed (see [18]), i.e. the following is true

$$\nabla_x E(F(x, \xi)) = E(\nabla_x F(x, \xi)). \tag{10}$$

Having this in mind, we can use the Law of large numbers again, and conclude that for every x almost surely

$$\lim_{N\to\infty} \nabla \hat{f}_N(x) = \nabla f(x).$$

This justifies using $\nabla \hat{f}_N(x)$ as an approximation of the measure of stationarity for problem (1). We have influence on that approximation because we can change the sample size N and, hopefully, make problem (4) closer to problem (1). Therefore (10), together with A1, helps us measure the performance of our algorithm regarding (1).

Having these assumptions in mind, one can easily prove the following three lemmas.

Lemma 2.1. *If A2 and A3 hold, then for every $x \in \mathbb{R}^n$ and every $N \in \mathbb{N}$ the following is true*

$$\|\nabla \hat{f}_N(x)\| \leq M_1.$$

Lemma 2.2. *If A2 is satisfied, then for every $N \in \mathbb{N}$ the function $\hat{f}_N$ is in $C^1(\mathbb{R}^n)$.*

Lemma 2.3. *If A4 holds, then for every $x \in \mathbb{R}^n$ and every $N \in \mathbb{N}$ the following is true*

$$M_F \leq \hat{f}_N(x) \leq M_{FF}.$$

We also state the following important lemma which, together with the previous two, guaranties that the line search is well defined.

Lemma 2.4. *[15] Suppose that function $h : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and let d_k be a descent direction for function h at point x_k. Also, suppose that h is bounded below on $\{x_k + \alpha d_k | \alpha > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist interval of step lengths satisfying the Wolf conditions (11) and (12)*

$$h(x_k + \alpha_k d_k) \leq h(x_k) + c_1 \alpha_k d_k^T \nabla h(x_k) \tag{11}$$

$$\nabla h(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla h(x_k)^T d_k \tag{12}$$

Backtracking technique that we use in order to find step size that satisfies the Armijo condition (11) will provide us with an α_k that satisfies the curvature condition (12) as well.

3 The Algorithm

The algorithm below is constructed to solve the problem (4) with the sample size N equal to some N_{max} which is observed as an input parameter. More precisely, we are searching for a stationary point of the function $\hat{f}_{N_{max}}$. The sample realization that defines the objective function $\hat{f}_{N_{max}}$ is generated at the beginning of optimization process. Therefore, we can say that the aim of the algorithm is to find a point x which satisfies

$$\|\nabla \hat{f}_{N_{max}}(x)\| = 0.$$

In this paper we assume that the suitable maximal sample size N_{max} can be determined without entering into details of such process.

As already stated the algorithm is constructed to let the sample size vary across the iterations and to let it decrease every time there is enough reason for that. Let us state the main algorithm here leaving the additional ones to be stated latter.

ALGORITHM 1.

S0 Input parameters: $N_{max}, N_0^{min} \in \mathbb{N},\ x_0 \in \mathbb{R}^n,\ \delta, \eta, \beta, \gamma_3, \nu_1 \in (0,1),\ \eta_0 < 1$.

S1 Generate the sample realization: $\xi_1, \ldots, \xi_{N_{max}}$.

Put $k = 0$, $N_k = N_0^{min}$.

S2 Compute $\hat{f}_{N_k}(x_k)$ and $\varepsilon_\delta^{N_k}(x_k)$ using (2) and (8).

S3 Test

If $\|\nabla \hat{f}_{N_k}(x_k)\| = 0$ and $N_k = N_{max}$ then STOP.

If $\|\nabla \hat{f}_{N_k}(x_k)\| = 0$, $N_k < N_{max}$ and $\varepsilon_\delta^{N_k}(x_k) > 0$ put $N_k = N_{max}$ and $N_k^{min} = N_{max}$ and go to step S2.

If $\|\nabla \hat{f}_{N_k}(x_k)\| = 0$, $N_k < N_{max}$ and $\varepsilon_\delta^{N_k}(x_k) = 0$ put $N_k = N_k + 1$ and $N_k^{min} = N_k^{min} + 1$ and go to step S2.

S4 Determine p_k such that $p_k^T \nabla \hat{f}_{N_k}(x_k) < 0$.

S5 Using the backtracking technique with the parameter β, find α_k such that

$$\hat{f}_{N_k}(x_k + \alpha_k p_k) \leq \hat{f}_{N_k}(x_k) + \eta \alpha_k p_k^T \nabla \hat{f}_{N_k}(x_k).$$

S6 Put $s_k = \alpha_k p_k, \quad x_{k+1} = x_k + s_k$ and compute dm_k using (7).

S7 Determine the candidate sample size N_k^+ using Algorithm 2.

S8 Determine the sample size N_{k+1} using Algorithm 3.

S9 Determine the lower bound of sample size N_{k+1}^{min}.

S10 Put $k = k + 1$ and go to step S2.

Before stating the auxiliary algorithms, let us briefly comment this one. The point x_0 is an arbitrary starting point. The sample realization generated in step S1 is the one that is used during the whole optimization process. For simplicity, if the required sample size is $N_k < N_{max}$, we take the first N_k realizations in order to calculate all relevant values. On the other hand, N_0^{min} is the lowest sample size that is going to be used in algorithm. The role of lower sample bound N_k^{min} will be clear after we state the remaining algorithms. The same is true for parameters η_0, γ_3 and ν_1.

Notice that the algorithm terminates after a finite number of iterations only if x_k is a stationary point of the function $\hat{f}_{N_{max}}$. Moreover, step S3 guaranties that we have a decreasing search direction in step S5, therefore the backtracking is well defined.

As we already mentioned, one of the main issues is how to determine the sample size that is going to be used in the next iteration. Algorithms 2 and 3 stated below provide details. As already mentioned Algorithm 2 is adopted from [2] and [3] to fit the line search framework and it leads us to the candidate sample size N_k^+. Acceptance of that candidate is decided within Algorithm 3. We will explain latter how to update N_k^{min}. For now, the important thing is that the lower bound is determined before we get to step S7 and it is considered as an input parameter in algorithm described below. Notice that the following algorithm is constructed to provide $N_k^{min} \leq N_k^+ \leq N_{max}$.

ALGORITHM 2.

S0 Input parameters: $dm_k,\ N_k^{min},\ \varepsilon_\delta^{N_k}(x_k),\ \nu_1 \in (0,1)$.

S1 Determine N_k^+

1) $dm_k = \varepsilon_\delta^{N_k}(x_k) \quad \rightarrow \quad N_k^+ = N_k$.

2) $dm_k > \varepsilon_\delta^{N_k}(x_k)$
Starting with $N = N_k$, while $dm_k > \varepsilon_\delta^{N}(x_k)$ and $N > N_k^{min}$, decrease N by 1 and calculate $\varepsilon_\delta^{N}(x_k) \quad \rightarrow \quad N_k^+$.

3) $dm_k < \varepsilon_\delta^{N_k}(x_k)$

- **i)** $dm_k \geq \nu_1 \varepsilon_\delta^{N_k}(x_k)$
Starting with $N = N_k$, while $dm_k < \varepsilon_\delta^{N}(x_k)$ and $N < N_{max}$, increase N by 1 and calculate $\varepsilon_\delta^{N}(x_k) \quad \rightarrow \quad N_k^+$.
- **ii)** $dm_k < \nu_1 \varepsilon_\delta^{N_k}(x_k) \quad \rightarrow \quad N_k^+ = N_{max}$.

The basic idea for this kind of reasoning can be found in [2] and [3]. The main idea is to compare two main measures of the progress, dm_k and $\varepsilon_\delta^{N_k}(x_k)$, and to keep them as close as possible to each other.

Let us consider dm_k as the benchmark. If $dm_k < \varepsilon_\delta^{N_k}(x_k)$, we say that $\varepsilon_\delta^{N_k}(x_k)$ is too big or that we have a lack of precision. That implies that the confidence interval is too wide and we are trying to narrow it down by increasing the sample size and therefore reducing the error made by approximation (2). On the other hand, in order to work with as small as possible sample size, if $dm_k > \varepsilon_\delta^{N_k}(x_k)$ we deduce that it is not necessary to have that much of precision and we are trying to reduce the sample size.

On the other hand, if we set the lack of precision as the benchmark, we have the following reasoning. If the reduction measure dm_k is too small (smaller than $\varepsilon_\delta^{N_k}(x_k)$), we say that there is not much that can be done for the function $\hat{f}_{N_k}$ in the sense of decreasing its value and we move on to the next level, trying to get closer to the final objective function $\hat{f}_{N_{max}}$ if possible.

Previously described mechanism provides us with the candidate for the upcoming sample size. Before accepting it, we have one more test. First of all, if the precision is increased, that is if $N_k \leq N_k^+$, we continue with $N_{k+1} = N_k^+$. However, if we have the signal that we should decrease the sample size, i.e. if $N_k^+ < N_k$, then we compare the reduction that is already obtained using the current step s_k and the sample size N_k with the reduction this step would provide if the sample size was N_k^+. In order to do that, we compute ρ_k using (9). If $\rho_k < \eta_0 < 1$, we do not approve the reduction because these two functions are too different and we choose to work with more precision and therefore put $N_{k+1} = N_k$. More formally, the algorithm is described as follows.

ALGORITHM 3.

S0 Input parameters: $N_k^+,\ N_k,\ x_k,\ x_{k+1},\ \eta_0 < 1$.

S1 Determine N_{k+1}

- **1)** If $N_k^+ \geqslant N_k$ then $N_{k+1} = N_k^+$.
- **2)** If $N_k^+ < N_k$ compute

$$\rho_k = \frac{\hat{f}_{N_k^+}(x_k) - \hat{f}_{N_k^+}(x_{k+1})}{\hat{f}_{N_k}(x_k) - \hat{f}_{N_k}(x_{k+1})}.$$

  - **i)** If $\rho_k \geqslant \eta_0$ put $N_{k+1} = N_k^+$.
  - **ii)** If $\rho_k < \eta_0$ put $N_{k+1} = N_k$.

Now we will describe how to update the lower bound N_k^{min}.

- If $N_{k+1} \leq N_k$ then $N_{k+1}^{min} = N_k^{min}$.
- If $N_{k+1} > N_k$ and
  - if N_{k+1} is a sample size which has not been used so far then $N_{k+1}^{min} = N_k^{min}$.
  - if N_{k+1} is a sample size which had been used and if we have made big enough decrease of the function $\hat{f}_{N_{k+1}}$ since the last time we used it, then $N_{k+1}^{min} = N_k^{min}$.
  - if N_{k+1} is a sample size which had been used and if we have not made big enough decrease of the function $\hat{f}_{N_{k+1}}$ since the last time we used it, then $N_{k+1}^{min} = N_{k+1}$.

We say that we have not made big enough decrease of the function $\hat{f}_{N_{k+1}}$ if for some constants $\gamma_3, \nu_1 \in (0,1)$ the following inequality is true

$$\hat{f}_{N_{k+1}}(x_{h(k)}) - \hat{f}_{N_{k+1}}(x_{k+1}) < \gamma_3 \nu_1 (k+1-h(k)) \varepsilon_\delta^{N_{k+1}}(x_{k+1}),$$

where $h(k)$ is the iteration at which we started to use the sample size N_{k+1} for the last time. For example, if $k=7$ and $(N_0, ..., N_8) = (3, 6, 6, 4, \mathbf{6}, 6, 3, 3, 6)$, then $N_k = 3$, $N_{k+1} = 6$ and $h(k) = 4$. So, the idea is that if we come back to some sample size N_{k+1} that we had already used and if, since then, we have not done much in order to decrease the value of $\hat{f}_{N_{k+1}}$ we choose not to go below that sample size anymore, i.e. we put it as the lower bound. At the end, notice that the sequence of the sample size lower bounds is nondecreasing.

4 Convergence analysis

This section is devoted to the convergence results for Algorithm 1. The following important lemma states that after a finite number of iterations the sample size becomes N_{max} and stays that way.

Lemma 4.1. *Suppose that assumptions A2 - A4 are true. Furthermore, suppose that there exist a positive constant κ and number $n_1 \in \mathbb{N}$ such that $\varepsilon_{\delta}^{N_k}(x_k) \geq \kappa$ for every $k \geq n_1$. Then, either Algorithm 1 terminates after a finite number of iterations with $N_k = N_{max}$ or there exists $q \in \mathbb{N}$ such that for every $k \geq q$ the sample size is maximal, i.e. $N_k = N_{max}$.*

Proof. First of all, recall that Algorithm 1 terminates only if $\|\nabla \hat{f}_{N_k}(x_k)\| = 0$ and $N_k = N_{max}$. Therefore, we will observe the case where the number of iterations is infinite. Notice that Algorithm 3 implies that $N_{k+1} \geq N_k^+$ is true for every k. Now, let us prove that sample size can not be stacked at a size that is lower than the maximal one.

Suppose that there exists $\tilde{n} > n_1$ such that for every $k \geq \tilde{n}$ $\quad N_k = N^1 < N_{max}$. We have already explained that step S3 of Algorithm 1 provides the decreasing search direction p_k at every iteration. Therefore, denoting $g_k^{N_k} = \nabla \hat{f}_{N_k}(x_k)$, we know that for every $k \geq \tilde{n}$

$$\hat{f}_{N^1}(x_{k+1}) \leq \hat{f}_{N^1}(x_k) + \eta \alpha_k (g_k^{N^1})^T p_k,$$

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

$$\begin{aligned}\hat{f}_{N^1}(x_{\tilde{n}+s}) &\leq \hat{f}_{N^1}(x_{\tilde{n}+s-1}) + \eta \alpha_{\tilde{n}+s-1} (g_{\tilde{n}+s-1}^{N^1})^T p_{\tilde{n}+s-1} \leq ... \\ &\leq \hat{f}_{N^1}(x_{\tilde{n}}) + \eta \sum_{j=0}^{s-1} \alpha_{\tilde{n}+j} (g_{\tilde{n}+j}^{N^1})^T p_{\tilde{n}+j}. \qquad (13)\end{aligned}$$

Now, from (13) and Lemma 2.3 we know that

$$-\eta \sum_{j=0}^{s-1} \alpha_{\tilde{n}+j} (g_{\tilde{n}+j}^{N^1})^T p_{\tilde{n}+j} \leq \hat{f}_{N^1}(x_{\tilde{n}}) - \hat{f}_{N^1}(x_{\tilde{n}+s}) \leq \hat{f}_{N^1}(x_{\tilde{n}}) - M_F. \qquad (14)$$

The inequality (14) is true for every s so

$$0 \leq \sum_{j=0}^{\infty} -\alpha_{\tilde{n}+j} (g_{\tilde{n}+j}^{N^1})^T p_{\tilde{n}+j} \leq \frac{\hat{f}_{N^1}(x_{\tilde{n}}) - M_F}{\eta} := C.$$

Therefore
$$\lim_{j\to\infty} -\alpha_{\tilde{n}+j}(\nabla \hat{f}_{N^1}(x_{\tilde{n}+j}))^T p_{\tilde{n}+j} = 0. \tag{15}$$

Let us observe the Algorithm 2 and iterations $k > \tilde{n}$. The possible scenarios are the following.

1) $dm_k = \varepsilon_\delta^{N_k}(x_k)$. This implies
$$-\alpha_k(g_k^{N_k})^T p_k = \varepsilon_\delta^{N_k}(x_k) \geq \kappa$$

2) $dm_k > \varepsilon_\delta^{N_k}(x_k)$. This implies
$$-\alpha_k(g_k^{N_k})^T p_k > \varepsilon_\delta^{N_k}(x_k) \geq \kappa$$

3) $dm_k < \varepsilon_\delta^{N_k}(x_k)$ and $dm_k \geq \nu_1 \varepsilon_\delta^{N_k}(x_k)$. In this case we have
$$-\alpha_k(g_k^{N_k})^T p_k \geq \nu_1 \varepsilon_\delta^{N_k}(x_k) \geq \nu_1 \kappa$$

4) The case $dm_k < \nu_1 \varepsilon_\delta^{N_k}(x_k)$ is impossible because it would yield $N_{k+1} \geq N_k^+ = N_{max} > N^1$.

Therefore, in every possible case we know that for every $k > \tilde{n}$
$$-\alpha_k(g_k^{N^1})^T p_k \geq \kappa\nu_1 := \tilde{C} > 0$$
and therefore
$$\liminf_{k\to\infty} -\alpha_k(g_k^{N^1})^T p_k \geq \tilde{C} > 0,$$
which is in contradiction with (15).

We have just proved that sample size can not stay on $N^1 < N_{max}$. Therefore, the remaining two possible scenarios are as follows:

L1 There exists $\tilde{n}$ such that for every $k \geq \tilde{n}$ $N_k = N_{max}$.

L2 The sequence of sample sizes oscillates.

Let us suppose that scenario L2 is the one that happens. Notice that this is the case where N_k^{min} can not reach N_{max} for any k. This is true because sequence of sample size lower bounds $\{N_k^{min}\}_{k\in\mathbb{N}}$ is nondecreasing and the existence of k such that $N_k^{min} = N_{max}$ would imply scenario L1. Therefore, for every k we know that

$$N_k^{min} < N_{max}.$$

Furthermore, this implies that the signal for increasing N_k^{min} could come only finitely many times, i.e. $N_{k+1}^{min} = N_{k+1}$ happens at most finitely many times because this case implies

$$N_{k+1}^{min} = N_{k+1} > N_k \geq N_{k-1}^{+} \geq N_{k-1}^{min}.$$

So, we conclude that there exists an iteration r such that for every $k \geq r$ we have one of the following scenarios:

M1 $N_{k+1} < N_k$

M2 $N_{k+1} > N_k$ and we have enough decrease in $\hat{f}_{N_{k+1}}$

M3 $N_{k+1} > N_k$ and we did not use the sample size N_{k+1} before

M4 $N_{k+1} = N_k$.

Now, let $\bar{N}$ be the maximal sample size that is used at infinitely many iterations. Furthermore, define the set of iterations $\bar{K_0}$ at which sample size changes to $\bar{N}$ and set $\bar{K} = \bar{K_0} \bigcap \{r, r+1, \ldots\}$. Notice that for every $k \in \bar{K}$

$$N_k < N_{k+1} = \bar{N}.$$

This implies that every iteration in $\bar{K}$ excludes the scenarios M1 and M4. Moreover, without loss of generality, we can say that scenario M3 is the one that can also be excluded. This leads us to the conclusion that M2 is the only possible scenario for iterations in $\bar{K}$. Therefore, for every $k \in \bar{K}$ the following is true

$$\hat{f}_{\bar{N}}(x_{h(k)}) - \hat{f}_{\bar{N}}(x_{k+1}) \geq \gamma_3 \nu_1 (k + 1 - h(k)) \varepsilon_{\delta}^{\bar{N}}(x_{k+1}).$$

Now, defining the set of iterations $K_1 = \bar{K} \bigcap \{n_1, n_1 + 1, \ldots\}$ we can say that for every $k \in K_1$ we have

$$\hat{f}_{\bar{N}}(x_{h(k)}) - \hat{f}_{\bar{N}}(x_{k+1}) \geq \gamma_3 \nu_1 \kappa > 0.$$

Recall that $h(k)$ defines the iteration at which we started to use the sample size $\bar{N}$ for the last time before the iteration $k + 1$. Therefore, previous

inequality implies that we have reduced the function $\hat{f}_{\bar{N}}$ for the positive constant $\gamma_3\nu_1\kappa$ infinitely many times, which is in contradiction with Lemma 2.3. From everything above, we conclude that the only possible scenario is in fact L1, i.e. there exists iteration $\tilde{n}$ such that for every $k \geq \tilde{n}, \quad N_k = N_{max}$.
■

Now, we will prove the main result. Before we state the theorem, we will make one more assumption about the search direction.

A5 The sequence of directions p_k generated at S4 of Algorithm 1 satisfies the following implication:

$$\lim_{k \in K} p_k^T \nabla \hat{f}_{N_k}(x_k) = 0 \; \Rightarrow \; \lim_{k \in K} \nabla \hat{f}_{N_k}(x_k) = 0,$$

for any subset of iterations K.

This assumption is obviously satisfied for $p_k = -\nabla \hat{f}_{N_k}(x_k)$.

Theorem 4.1. *Suppose that assumptions A2 - A5 are true. Furthermore, suppose that there exist a positive constant κ and number $n_1 \in \mathbb{N}$ such that $\varepsilon_{\delta}^{N_k}(x_k) \geq \kappa$ for every $k \geq n_1$ and that the sequence $\{x_k\}_{k \in \mathbb{N}}$ generated by Algorithm 1 is bounded. Then, either Algorithm 1 terminates after a finite number of iterations at a stationary point of function $\hat{f}_{N_{max}}$ or every accumulation point of the sequence $\{x_k\}_{k \in \mathbb{N}}$ is a stationary point of $\hat{f}_{N_{max}}$.*

Proof. First of all, recall that Algorithm 1 terminates only if $\|\nabla \hat{f}_{N_{max}}(x_k)\| = 0$, that is if the point x_k is stationary for the function $\hat{f}_{N_{max}}$. Therefore, we will observe the case where the number of iterations is infinite. In that case, the construction of Algorithm 1 provides us with a decreasing search direction at every iteration. Furthermore, Lemma 4.1 implies the existence of iteration $\hat{n}$ such that for every $k \geq \hat{n} \quad N_k = N_{max}$ and

$$\hat{f}_{N_{max}}(x_{k+1}) \leq \hat{f}_{N_{max}}(x_k) + \eta \alpha_k (g_k^{N_{max}})^T p_k,$$

where $g_k^{N_{max}} = \nabla \hat{f}_{N_{max}}(x_k)$. Equivalently, for every $s \in \mathbb{N}$

$$\hat{f}_{N_{max}}(x_{\hat{n}+s}) \leq \hat{f}_{N_{max}}(x_{\hat{n}+s-1}) + \eta \alpha_{\hat{n}+s-1} (g_{\hat{n}+s-1}^{N_{max}})^T p_{\hat{n}+s-1} \leq \ldots$$

$$\leq \hat{f}_{N_{max}}(x_{\hat{n}}) + \eta \sum_{j=0}^{s-1} \alpha_{\hat{n}+j} (g_{\hat{n}+j}^{N_{max}})^T p_{\hat{n}+j}.$$

Again, this inequality and Lemma 2.3 imply

$$-\eta \sum_{j=0}^{s-1} \alpha_{\hat{n}+j} (g_{\hat{n}+j}^{N_{max}})^T p_{\hat{n}+j} \leq \hat{f}_{N_{max}}(x_{\hat{n}}) - \hat{f}_{N_{max}}(x_{\hat{n}+s}) \leq \hat{f}_{N_{max}}(x_{\hat{n}}) - M_F.$$

This is true for every $s \in \mathbb{N}$, therefore

$$0 \leq \sum_{j=0}^{\infty} -\alpha_{\hat{n}+j} (g_{\hat{n}+j}^{N_{max}})^T p_{\hat{n}+j} \leq \frac{\hat{f}_{N_{max}}(x_{\hat{n}}) - M_F}{\eta} := C.$$

This implies that

$$\lim_{j \to \infty} -\alpha_{\hat{n}+j} (\nabla \hat{f}_{N_{max}}(x_{\hat{n}+j}))^T p_{\hat{n}+j} = 0. \tag{16}$$

We will prove that

$$\lim_{k \to \infty} -(\nabla \hat{f}_{N_{max}}(x_k))^T p_k = 0. \tag{17}$$

Suppose the contrary, i.e. suppose that there exists a positive constant M and a subset of iterations K such that for every $k \in K_1 = K \cap \{\hat{n}, \hat{n}+1, ...\}$

$$-(\nabla \hat{f}_{N_{max}}(x_k))^T p_k \geq M > 0.$$

In that case, (16) implies that $\lim_{k \in K_1} \alpha_k = 0$. Therefore, there exists $\hat{k}$ such that for every $k \in K_2 = K_1 \cap \{\hat{k}, \hat{k}+1, ...\}$ the step size α_k that satisfies the Armijo condition (6) is smaller than 1. That means that for every $k \in K_2$ there exists α'_k such that $\alpha_k = \beta \alpha'_k$ and

$$\hat{f}_{N_{max}}(x_k + \alpha'_k p_k) > \hat{f}_{N_{max}}(x_k) + \eta \alpha'_k (\nabla \hat{f}_{N_{max}}(x_k))^T p_k,$$

which is equivalent to

$$\frac{\hat{f}_{N_{max}}(x_k + \alpha'_k p_k) - \hat{f}_{N_{max}}(x_k)}{\alpha'_k} > \eta (\nabla \hat{f}_{N_{max}}(x_k))^T p_k. \tag{18}$$

Notice that $\lim_{k \in K_2} \alpha'_k = 0$. Taking the limit in (18) and using Lemma 2.2, we obtain

$$(\nabla \hat{f}_{N_{max}}(x_k))^T p_k \geq \eta (\nabla \hat{f}_{N_{max}}(x_k))^T p_k. \tag{19}$$

On the other hand, we know that $\eta \in (0,1)$ and p_k is decreasing direction, i.e. $(\nabla \hat{f}_{N_{max}}(x_k))^T p_k < 0$. This implies that

$$(\nabla \hat{f}_{N_{max}}(x_k))^T p_k < \eta (\nabla \hat{f}_{N_{max}}(x_k))^T p_k,$$

which is in obvious contradiction with (19). This leads us to the conclusion that (17) must be true. Now, assumption A5 implies that

$$\lim_{k\to\infty} \nabla \hat{f}_{N_{max}}(x_k) = 0.$$

Notice that, since the sequence of iterations $\{x_k\}_{k\in\mathbb{N}}$ is bounded, we know that there exists at least one accumulation point of that sequence. Let x^* be an arbitrary accumulation point of $\{x_k\}_{k\in\mathbb{N}}$,

$$\lim_{j\to\infty} x_{k_j} = x^*.$$

Finally, using Lemma 2.2 we conclude that

$$0 = \lim_{k\to\infty} \nabla \hat{f}_{N_{max}}(x_k) = \lim_{j\to\infty} \nabla \hat{f}_{N_{max}}(x_{k_j}) = \nabla \hat{f}_{N_{max}}(\lim_{j\to\infty} x_{k_j}) = \nabla \hat{f}_{N_{max}}(x^*).$$

We have just proved that every accumulation point of the sequence $\{x_k\}_{k\in\mathbb{N}}$ is a stationary point of function $\hat{f}_{N_{max}}$. This completes the proof ■

5 Numerical implementation

In this section, we are going to present some numerical results obtained by Algorithm 1. The first subsection contains the results obtained on two academic test examples while the second subsection deals with the discrete choice problem that is relevant in many applications. The test examples presented in 5.1 are Allufi - Pentini's [13] and Rosenbrock problem [6] in noisy environment. Both of them are convenient for initial testing purposes as one can solve them analytically and thus we can actually compute some quality indicators of the approximate solutions obtained by the presented variable sample size line search methods. One of the assumptions in this paper is that the analytical form of the gradient $\nabla_x F(x,\xi)$ is available. This assumption is not satisfied in many real applications. We have used some of the test examples with gradient approximations in order to check the applicability of the presented algorithm in cases where the analytical gradient is unavailable. The Mixed Logit problem is slightly different than the problem (4). Given the practical importance of this problem we introduce minor adjustments of Algorithm 1 and report the results in 5.2.

Algorithm 1 uses an unspecified descent direction p_k at step S4. We report the results for two possible directions, the steepest descent direction for $\hat{f}_{N_k}$,

$$p_k = -\nabla \hat{f}_{N_k}(x_k), \tag{20}$$

and the second order direction obtained by

$$p_k = -H_k \nabla \hat{f}_{N_k}(x_k), \tag{21}$$

where H_k is a positive definite matrix that approximates the inverse Hessian matrix $(\nabla^2 \hat{f}_{N_k}(x_k))^{-1}$. Among many options for H_k we have chosen the BFGS approach. We also let $H_0 = I$ where I denotes the identity matrix. Other possibilities for the initial approximation H_0 can be seen in [15] and [19]. The inverse Hessian approximation is updated by the famous BFGS formula that can be found in [15]. More precisely, we compute s_k as in step S6 of Algorithm 1 and let

$$y_k = \nabla \hat{f}_{N_{k+1}}(x_{k+1}) - \nabla \hat{f}_{N_k}(x_k).$$

We compute y_k after step S8 when the next iteration x_{k+1} and the relevant sample size N_{k+1} are determined. Then, if $y_k^T s_k > 0$, we use BFGS update formula to obtain

$$H_{k+1} = (I - \frac{s_k y_k^T}{y_k^T s_k}) H_k (I - \frac{y_k s_k^T}{y_k^T s_k}) + \frac{s_k s_k^T}{y_k s_k^T}.$$

Otherwise, we put $H_{k+1} = H_k$. This way we can be sure that our approximation matrix remains positive definite, therefore providing the decreasing search direction (21).

Notice also that the assumption A5 is satisfied for both direction (20) or (21), but in the case of (21) we need to assume that $F(\cdot, \xi) \in C^2$ instead of A2. Furthermore, some kind of boundedness for H_k is also necessary. BFGS matrix in noisy environment is analyzed in [11].

If we choose to apply the safeguard rule presented in Algorithm 3, we set the input parameter η_0 to be some finite number smaller than 1. On the other hand, if we set $\eta_0 = -\infty$ the safeguard rule is not applied and thus the algorithm accepts the candidate sample size for the next iteration. In other words, for every iteration k we have that $N_{k+1} = N_k^+$.

Based on the descent direction choice and the safeguard rule application, four different implementations of Algorithm 1 are to be specified. NG represents the algorithm that uses negative gradient search directions (20) without the safeguard rule i.e. with $\eta_0 = -\infty$, while NG - ρ uses the negative gradient direction and enforces the safeguard rule. Analogously, BFGS stands for second order type directions (21) with BFGS - ρ being the algorithm with the safeguard rule. All of them are tested in the following subsections.

5.1 Numerical results for noisy problems

First of all, we are going to present numerical results obtained by applying Algorithm 1 to Aluffi - Pentini's problem which can be found in [13]. Originally, this is deterministic problem with box constraints. Following the ideas from [6], we added the noise to the first component of decision variable and removed the constraints, so the objective function becomes

$$f(x) = E(0.25(x_1\xi)^4 - 0.5(x_1\xi)^2 + 0.1\xi x_1 + 0.5x_2^2),$$

where ξ represents a random variable with Normal distribution

$$\xi : \mathcal{N}(1, \sigma^2). \tag{22}$$

We observed this problem with three different levels of variance. As we are able to calculate the real objective function and analytical form of its gradient, we can actually see how close are the approximate and the true stationary points. Table 1 contains the stationary points for various levels of noise and the global minimums of the relevant objective functions.

σ^2	global minimizer - x^*	local minimizer	maximizer	$f(x^*)$
0.01	$(-1.02217, 0)$	$(0.922107, 0)$	$(0.100062, 0)$	-0.340482
0.1	$(-0.863645, 0)$	$(0.771579, 0)$	$(0.092065, 0)$	-0.269891
1	$(-0.470382, 0)$	$(0.419732, 0)$	$(0.05065, 0)$	-0.145908

Table 1: Stationary points for Allufi - Pentini's problem

The parameters are set as follows. The stopping criterion is

$$\|\nabla \hat{f}_{N_{max}}(x_k)\| < 10^{-2}.$$

The initial sample size is set to be $N_0^{min} = 3$, the Armijo parameter $\eta = 10^{-4}$, while the confidence level is $\delta = 0.95$. The backtracking is performed using $\beta = 0.5$ and the input parameters for Algorithm 2 are

$$\nu_1 = \frac{1}{\sqrt{N_{max}}} \text{ and } \gamma_3 = 0.5.$$

The safeguard parameter in algorithms NG - ρ and BFGS - ρ is $\eta_0 = 0.7$, while the initial approximation is $x_0 = (1, 1)^T$. These parameters are the same for all levels of noise.

We conducted 50 independent runs of each algorithm. The sample of size N_{max} is generated for each run and all algorithms are tested with that same sample realization. The results in the following tables are the average values obtained from these 50 runs. Columns $\|\nabla \hat{f}_{N_{max}}\|$ and $\|\nabla f\|$ give, respectively, the average values of the gradient for the approximate problem (4) and the initial problem (1) objective function at the last iteration, while fev represents the average number of function evaluations with one gradient evaluation being counted as n function evaluations. The final column $fevNmax$ represents the average number of function evaluations when $N_k = N_{max}$ at every iteration of algorithm i.e. the cost of the sample path method.

$\sigma^2 = 0.01$, $N_{max} = 100$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
NG	0.008076	0.014906	1402	1868
NG - ρ	0.008002	0.013423	1286	
BFGS	0.003575	0.011724	840	928
BFGS - ρ	0.003556	0.012158	793	
$\sigma^2 = 0.1$, $N_{max} = 200$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
NG	0.007545	0.027929	3971	4700
NG - ρ	0.006952	0.028941	3537	
BFGS	0.003414	0.027991	2155	2968
BFGS - ρ	0.003879	0.027785	2152	
$\sigma^2 = 1$, $N_{max} = 600$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
NG	0.006072	0.050208	13731	15444
NG - ρ	0.005149	0.058036	10949	
BFGS	0.003712	0.054871	7829	14760
BFGS - ρ	0.002881	0.055523	8372	

Table 2: Allufi - Pentini's problem

As expected, the results in Table 2 confirm that the variable sample size strategy is significantly cheaper than the sample path line search with the maximal sample size. At the same time, given that N_{max} is eventually reached the approximate solutions obtained by the four tested methods are of the same quality as the sample path solutions i.e. they are the stationary points of $\hat{f}_{N_{max}}$. One can also notice that the algorithms that use the second order search directions performed better than their negative gradient counterparts. The application of the safeguard rule decreases the cost in all tested cases except in the last case with the highest variance and second order search direction.

Given that the considered problems have more than one stationary point we report the distribution of the approximate stationary points in Table

3. Column *global* counts how many times we had convergence towards the global minimizer, column *local* shows how many replications converged to the local minimizer and column *max* counts convergence to the stationary point that is the maximizer of objective function f. Columns fgm and flm represent the average values of function f in the runs that converged to the global minimizer and local minimizer, respectively.

$\sigma^2 = 0.01$, $N_{max} = 100$					
Algorithm	*global*	*local*	*max*	fgm	flm
NG	0	50	0	-	-0.14543
NG - ρ	0	50	0	-	-0.14545
BFGS	0	50	0	-	-0.14546
BFGS - ρ	0	50	0	-	-0.14545
$\sigma^2 = 0.1$, $N_{max} = 200$					
Algorithm	*global*	*local*	*max*	fgm	flm
NG	11	39	0	-0.26940	-0.10562
NG - ρ	15	35	0	-0.26940	-0.10563
BFGS	12	38	0	-0.26948	-0.10559
BFGS - ρ	12	38	0	-0.26946	-0.10560
$\sigma^2 = 1$, $N_{max} = 600$					
Algorithm	*global*	*local*	*max*	fgm	flm
NG	28	19	3	-0.14537	-0.05625
NG - ρ	35	15	0	-0.14529	-0.05626
BFGS	30	19	1	-0.14537	-0.05612
BFGS - ρ	31	19	0	-0.14538	-0.05613

Table 3: The approximate stationary points for Allufi - Pentini's problem

Notice that as the variance increases, the number of replications that are converging towards global minimizers increases as well. However, we also registered convergence towards maximizers when we increased the variance.

The next example relays on Rosenbrock function. Following the example from [6], we added the noise to the first component in order to make it random. We obtained the following objective function

$$f(x) = E(100(x_2 - (x_1\xi)^2)^2 + (x_1\xi - 1)^2), \tag{23}$$

where ξ is the random variable defined with (22). This kind of function has only one stationary point which is global minimizer, but it depends on level of noise. The algorithms are tested with the dispersion parameter σ^2 equal to 0.001, 0.01 and 0.1. An interesting observation regarding this problem is that the objective function (23) becomes more and more "optimization friendly" when the variance increases. Therefore, we put the same maximal sample size for all levels of noise. The stationary points and the minimal values of the objective function are given in Table 4.

σ^2	global minimizer - x^*	$f(x^*)$
0.001	$(0.711273, 0.506415)$	0.186298
0.01	$(0.416199, 0.174953)$	0.463179
0.1	$(0.209267, 0.048172)$	0.634960

Table 4: Rosenbrock problem - the global minimizers

Minimization of the Rosenbrock function is a well known problem and in general the second order directions are necessary to solve it. The same appears to be true in noisy environment. As almost all runs with the negative gradient failed, only BFGS type results are presented in Table 5. All the parameters are the same as in the previous example except that the initial iteration is set to be $x_0 = (-1, 1.2)^T$.

$\sigma^2 = 0.001$, $N_{max} = 3500$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
BFGS	0.003413	0.137890	56857	246260
BFGS - ρ	0.003068	0.137810	49734	
$\sigma^2 = 0.01$, $N_{max} = 3500$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
BFGS	0.002892	0.114680	56189	213220
BFGS - ρ	0.003542	0.114160	52875	
$\sigma^2 = 0.1$, $N_{max} = 3500$				
Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\|\nabla f\|$	fev	$fevNmax$
BFGS	0.003767	0.093363	67442	159460
BFGS - ρ	0.003561	0.093290	59276	

Table 5: Rosenbrock problem

The same conclusion is valid for this example as for Aluffi - Pentini's problem - the variable sample size strategy reduces the number of function evaluations. Moreover, as far as this example is concerned, clear advantage is assigned to the algorithm that uses the safeguard rule.

So far we have compared the number of function evaluations for algorithms proposed in this paper and the sample path algorithm with N_{max} sample size. The existing methods include the line search sample path methods where the sample size is increasing during the iterative process. Such methods in general start with a modest sample size and increase it as the iterates progress. Therefore the natural question here is how different is the strategy advocated in this paper i.e. how often do we have decrease in the sample size. The percentage of iterations in which decrease of the sample size occurs varies across the problems and algorithms. It depends both on noise level and search direction. Observing the average of 50 runs in the previous two examples the percentage of the decreasing sample size iterations varies between 11% and 32%. The lowest number occurred in Aluffi - Pentini's problem with $\sigma^2 = 0.01$ and BFGS - ρ algorithm while the highest one was detected when BFGS - ρ algorithm was applied on Rosenbrock function with the highest tested noise level.

Percentage of iterations where the decrease of a sample size was rejected due to safeguard rule from Algorithm 3 also differs. In Rosenbrock problem it was approximately in 25% of cases for all levels of noise, while in Aluffi - Pentini's problem it varied more. The lowest percentage occurred in case of BFGS - ρ algorithm with $\sigma^2 = 0.1$ and it was 32%. The highest one was 66% and it was detected in case where variance was equal to 1 and the negative gradient search direction was used. In this particular case, comparing the algorithm with and without safeguard rule, the greatest decrease in number of function evaluations, around 20%, was obtained as well. The results thus clearly indicate that the decrease in sample size is happening frequently enough to lower the number of function evaluations. Also, ρ type algorithms clearly outperform the algorithms without the safeguard in almost all tested cases and thus this rule indeed prevents at least some of the unproductive sample decreases. Clearly, the conclusions and comments presented here are influenced by the considered examples and more testing is needed to establish the optimal values of all parameters that are used.

Let us now recall that one of the assumptions in this paper is that the analytical form of gradient $\nabla_x F(x, \xi)$ is available. This assumption is not too realistic as the gradient is unavailable very often. This is the case, for example, when we are dealing with black-box mechanisms. There are also various examples where simulations are used in order to approximate the value of the objective function. Therefore, it seems important to investigate the behavior of the proposed algorithms if the true gradient is unavailable. The approximate gradient values in noisy environment are complicated issue and a lot of research is devoted to that topic, see [1, 8, 9, 12, 19]. We will present some initial results obtained with two types of gradient approximation for Allufi - Pentini's problem. The purpose of these results is to demonstrate that the algorithms could be used even if the analytical gradient is not available. However, further research is needed for any kind of conclusion regarding the choice of gradient approximation. We tested two of the well know approximation methods. First we consider the finite difference technique to approximate the gradient in optimization process. More precisely, we used the central (two-sided symmetric) difference gradient estimator for each component of the gradient function. The ith component of the gradient is therefore approximated by

$$\nabla_{x_i} \hat{f}_N(x) \approx \frac{\hat{f}_N(x + he_i) - \hat{f}_N(x - he_i)}{2h}, \tag{24}$$

where e_i is the ith column of identity matrix. In our example, parameter h is set to be 10^{-4}. This is a special case of an estimator that can be found for example, in [8]. One can see in the same paper some other methods for gradient estimation as well. Some tests were performed with the one sided finite difference estimator which is cheaper but they did not provide satisfactory results and they are not reported.

The second gradient approximation we tested are the simultaneous perturbations estimators. These gradient estimators can also be found in [8] and their main advantage is that they require only 2 evaluations of function $\hat{f}_N$ regardless of the problem dimension. The first one that we used is

$$\nabla_{x_i} \hat{f}_N(x) \approx \frac{\hat{f}_N(x + h\Delta) - \hat{f}_N(x - h\Delta)}{2h\Delta_i}, \tag{25}$$

where $\Delta = (\Delta_1, ..., \Delta_n)^T$ is a vector whose components are i.i.d. random variables with mean zero and finite inverse second moment. We specified the distribution for each Δ_i to be symmetric Bernoulli i.e. Δ_i can take values 1 and -1, both with probability 0.5. Parameter h was like in finite difference estimator case. Unfortunately, this estimator did not provide satisfactory results. On the other hand, the similar estimator from [8] performed much better. Its form is slightly different from (25), but it permits the usage of a Normal distribution for perturbation sequence. We specified it to be

$$\nabla_{x_i} \hat{f}_N(x) \approx \frac{(\hat{f}_N(x + h\Delta) - \hat{f}_N(x - h\Delta))\Delta_i}{2h}, \tag{26}$$

where $h = 10^{-4}$ and each Δ_i follows standardized Normal distribution.

Retaining all the other parameters as in previous testings, we obtained the following results for Aluffi - Pentini's problem. First part of the Table 6 corresponds to the finite difference estimator (FD) given by (24), while the remaining one refers to the simultaneous perturbations estimator (SP) defined by (26). In the following table column g represents the average value of the corresponding gradient estimators.

	FD			SP		
$\sigma^2 = 0.01$, $N_{max} = 100$						
Algorithm	g	$\|\nabla f\|$	fev	g	$\|\nabla f\|$	fev
NG	0.008076	0.014906	2632	0.003578	0.091941	1903
NG - ρ	0.008003	0.013423	2423	0.003514	0.102590	1850
BFGS	0.003575	0.011724	1504	0.004309	0.284110	8240
BFGS - ρ	0.003556	0.012158	1431	0.005287	0.286170	8402
$\sigma^2 = 0.1$, $N_{max} = 200$						
Algorithm	g	$\|\nabla f\|$	fev	g	$\|\nabla f\|$	fev
NG	0.007545	0.027929	7341	0.003684	0.069363	5147
NG - ρ	0.006952	0.028941	6504	0.003265	0.121340	4432
BFGS	0.003414	0.027991	3863	0.004889	0.350460	14715
BFGS - ρ	0.003879	0.027786	3858	0.004617	0.310260	9102
$\sigma^2 = 1$, $N_{max} = 600$						
Algorithm	g	$\|\nabla f\|$	fev	g	$\|\nabla f\|$	fev
NG	0.006072	0.050209	21391	0.004383	0.126370	22796
NG - ρ	0.005149	0.058036	17352	0.004516	0.163630	17202
BFGS	0.003685	0.054723	14289	0.004861	1.079000	62671
BFGS - ρ	0.002880	0.055522	15351	0.005141	1.151800	62779

Table 6: Gradient approximation algorithms for Allufi - Pentini's problem

The application of gradient approximations yielded significantly weaker results but nevertheless FD approximation seems to generate reasonably good approximations of the stationary points. The distribution of approximate stationary points is quite similar to the distribution presented in Table 3. Roughly speaking, the number of function evaluations is twice as large as the number of evaluations with analytical gradient estimator $\nabla \hat{f}_N$. The values of $\|\nabla f\|$ are reasonably small given the stopping criteria and the algorithms were successful in all runs. The algorithms that use BFGS directions were cheaper than the ones with the negative gradient approximations and the safeguard rule application saved some function evaluations. On the other hand, SP approximation performed significantly worse as it approximates the real gradient quite poorly. The number of function evaluations was significantly smaller with NG type algorithms if compared with FD approach but the quality of approximate solutions is also worse. The BFGS algorithms were clearly outperformed by NG algorithms which is consistent with poor quality of gradient estimation in SP approach. In this case, the second order direction is in fact worse than the first order direction as the errors propagate. SP algorithms were also rather unstable if we consider the distribution of achieved stationary points. The number of runs which resulted with global minimizers increased but the number of runs in which algorithms were converging towards maximizers is larger too. Furthermore some of the 50 runs were unsuccessful within 500000 function evaluations - one run of BFGS algorithm with $\sigma^2 = 0.1$ and one run of BFGS - ρ for $\sigma^2 = 1$. NG algorithm did not manage to converge in 8 runs with the largest variance level. The results presented in Table 6 are the average values of the successful runs only.

In general, the problem of finding the suitable gradient estimator is very tempting. As we already mentioned, some of the gradient free approaches, see for example [5, 7], will probably make these algorithms more applicable in practice. Therefore, it will be the subject of future research.

5.2 Application to discrete choice theory - Mixed Logit models

In this section we are going to present numerical results obtained by applying slightly modified algorithms on simulated data. That data represent real world problems that come from the discrete choice theory. Discrete choice problems are subject of various disciplines like econometrics, transportation, psychology etc. The problem that will be considered is an unconstrained parameter estimation problem. We will briefly describe the problem while the more detailed description with further references can be found in [2, 3, 4].

Let us consider a set of r_a agents and r_m alternatives. Suppose that every agent chooses one of finitely many alternatives. The choice is made according to r_k characteristics that each alternative has. Suppose that they are all numerical. Further, each agent chooses the alternative that maximizes his utility. Utility of agent i for alternative j is given by

$$U_{i,j} = V_{i,j} + \varepsilon_{i,j},$$

where $V_{i,j}$ depends on the vector of characteristics of alternative j ($m_j = (k_1^j, ..., k_{r_k}^j)^T$) and $\varepsilon_{i,j}$ is the error term. We will observe probably the most popular model in practice where $V_{i,j}$ is a linear function, that is

$$V_{i,j} = V_{i,j}(\beta^i) = m_j^T \beta^i.$$

We specified β^i, $i = 1, 2, ..., r_a$ to be a vector with r_k Normally distributed components. More precisely,

$$\beta^i = (\beta_1^i, ..., \beta_{r_k}^i)^T = (\mu_1 + \xi_1^i \sigma_1, ..., \mu_{r_k} + \xi_{r_k}^i \sigma_{r_k})^T,$$

where ξ_j^i, $i = 1, 2, ..., r_a$, $j = 1, 2, ..., r_k$ are i.i.d. random variables with standardized Normal distribution. In other words, $\beta_k^i : \mathcal{N}(\mu_k, \sigma_k^2)$ for every i. The parameters μ_k and σ_k, $k = 1, 2, ..., r_k$ are the ones that we are trying to estimate. Therefore, they will constitute the vector x of unknowns and the dimension of our problem is going to be $n = 2r_k$. The term $\varepsilon_{i,j}$ is a random variable whose role is to collect all the factors that are not included in the function $V_{i,j}$. It can also be viewed as the taste of each agent. Different assumptions about these terms lead to different models. We will assume that for every i and every j the random variable $\varepsilon_{i,j}$ follows Gumbel distribution with mean 0 and scale parameter 1. Gumbel distribution is also known as type 1 extreme value distribution.

Now, suppose that every agent made his own choice among these alternatives. The problem we want to solve is to maximize the likelihood function. Under the assumptions that we made, if the realization $\bar{\xi}^i$ of $\xi^i = (\xi_1^i, ..., \xi_{r_k}^i)^T$ is known, the probability that agent i chooses alternative j becomes

$$L_{i,j}(x, \bar{\xi}^i) = \frac{e^{V_{i,j}(x,\bar{\xi}^i)}}{\sum_{s=1}^{r_m} e^{V_{i,s}(x,\bar{\xi}^i)}}.$$

Moreover, the unconditional probability is therefore given by

$$P_{i,j}(x) = E(L_{i,j}(x, \xi^i)).$$

Now, if we denote by $j(i)$ the choice of agent i, the problem becomes

$$\max_{x \in \mathbb{R}^n} \prod_{i=1}^{r_a} P_{i,j(i)}(x). \tag{27}$$

The equivalent form of (27) is given by

$$\min_{x \in \mathbb{R}^n} -\frac{1}{r_a} \sum_{i=1}^{r_a} \ln E(L_{i,j(i)}(x, \xi^i)).$$

Notice that this problem is similar, but not exactly the same as (1). The objective function is now

$$f(x) = -\frac{1}{r_a}\sum_{i=1}^{r_a} \ln E(L_{i,j(i)}(x,\xi^i)),$$

so the approximating function will be

$$\hat{f}_N(x) = -\frac{1}{r_a}\sum_{i=1}^{r_a} \ln(\frac{1}{N}\sum_{s=1}^{N} L_{i,j(i)}(x,\xi_s^i)).$$

Here $\xi_1^i, ..., \xi_N^i$ are independent realizations of the random vector ξ^i. The realizations are independent across the agents as well. Notice that it is not too difficult to calculate the exact gradient of $\hat{f}_N$. Therefore, we have the problem where derivative-based approach is suitable.

One of the main differences between algorithms presented in previous sections and the ones that are used for Mixed Logit problem is the way that we calculate the "lack of precision" $\varepsilon_\delta^N(x)$. We will define the approximation of the confidence interval radius just like it is proposed in [4],

$$\varepsilon_\delta^N(x) = \frac{\alpha_\delta}{r_a}\sqrt{\sum_{i=1}^{r_a}\frac{\hat{\sigma}^2_{N,i,j(i)}(x)}{NP^2_{i,j(i)}(x)}}. \tag{28}$$

Here, α_δ represents the same parameter as in (8) and $\hat{\sigma}^2_{N,i,j(i)}(x)$ is the sample variance estimator, i.e.

$$\hat{\sigma}^2_{N,i,j(i)}(x) = \frac{1}{N-1}\sum_{s=1}^{N}(L_{i,j(i)}(x,\xi_s^i) - \frac{1}{N}\sum_{k=1}^{N}(L_{i,j(i)}(x,\xi_k^i))^2.$$

Confidence level that is used for numerical testings is retained at 0.95, therefore $\alpha_\delta \approx 1.96$. The reason for taking (28) is the fact that it can be shown, by using Delta method [17, 18], that in this case $\sqrt{N}(f(x) - \hat{f}_N(x))$ converges in distribution towards random variable with Normal distribution with mean zero and variance equal to $\frac{1}{N^2}\sum_{i=1}^{r_a}\frac{\sigma^2_{i,j(i)}(x)}{P^2_{i,j(i)}(x)}$.

Let us briefly analyze the convergence conditions for the adjusted algorithm. First of all, notice that for every N, function $\hat{f}_N$ is nonnegative and thus the lower bound in Lemma 2.3 is zero. Assumptions A2, A3 and A4 can be reformulated in a following way

B2 For every N, $\quad \hat{f}_N \in C^1(\mathbb{R}^n)$.

B3 There is a positive constant M_1 such that for every N, x, $\|\nabla \hat{f}_N(x)\| \leq M_1$.

B4 There exists positive constant M_{FF} such that for every N, x, $\quad \hat{f}_N(x) \leq M_{FF}$.

The following result holds.

Theorem 5.1. *Suppose that B2 - B4 and A5 are satisfied. Furthermore, suppose that there exist a positive constant* κ *and number* $n_1 \in \mathbb{N}$ *such that* $\varepsilon_\delta^{N_k}(x_k) \geq \kappa$ *for every* $k \geq n_1$ *and that the sequence* $\{x_k\}_{k\in\mathbb{N}}$ *generated by the adjusted Algorithm 1 is bounded. Then, either the adjusted Algorithm 1 terminates after a finite number of iterations at a stationary point of* $\hat{f}_{N_{max}}$ *or every accumulation point of the sequence* $\{x_k\}_{k\in\mathbb{N}}$ *is a stationary point of* $\hat{f}_{N_{max}}$.

The test problem is generated as follows. The number of alternatives and characteristics is 5. Therefore, we generated a matrix M of size 5×5 using the standardized Normal distribution. Each column of that matrix represents the characteristics for one of the alternatives. The number of agents is assumed to be 500. Furthermore, we generated a matrix B with 5 rows and 500 columns where ith column represents the realization of random vector β^i. More precisely, we set each component of matrix B to be a realization of Normally distributed random variable with mean 0.5 and variance 1, i.e. $B(i,j) : \mathcal{N}(0.5, 1)$. At the end, we formed a matrix of random terms $\varepsilon_{i,j}$ with 5 rows and 500 columns. Each component of that matrix is a realization of Gumbel distribution with mean 0 and scale parameter 1. We used these matrices to find the vector of choices for the agents.

The results presented in Table 7 are obtained after 10 independent runs of each algorithm, including the algorithms with fixed sample size. At each run, the initial iteration is set to be $x_0 = (0.1, \ldots, 0.1)^T$. The maximal sample size for each agent is $N_{max} = 500$. Since we use independent samples across the agents, the total maximal sample size is 250000. Thus, this is the number of realizations of random vector ξ which are generated at the beginning of the optimization process. In algorithms with variable sample size, the starting sample size for each agent is $N_0^{min} = 3$. The other parameters are set as in the previous subsection.

Since this is a real world problem which can hardly be solved without some numerical algorithm, we are not able to calculate the value of the true objective function nor the gradient at any point. Therefore, we used sample of size $N = 2000$ to approximate the true value of the gradient. Empirically, we noticed that sample size 2000 did not yield much discrepancy from sample size 10000. The average values of $\nabla \hat{f}_{2000}$ at the final iterations are presented in column $\tilde{g}$ of Table 7. The remaining notation is just like in previous subsection.

Algorithm	$\|\nabla \hat{f}_{N_{max}}\|$	$\tilde{g}$	fev	$fevNmax$
NG	0.008888	0.008101	4.4668E+07	9.5300E+07
NG - ρ	0.009237	0.008530	3.8611E+07	
BFGS	0.004128	0.003498	6.2430E+06	1.7750E+07
BFGS - ρ	0.004616	0.004256	5.7895E+06	

Table 7 : Mixed Logit Problem

According to fev columns, the algorithms with variable sample size strategy once again did better than their fixed-size counterparts. This is even more obvious in the BFGS search direction cases. Moreover, these algorithms clearly outperformed their negative gradient competitors. Notice also that the safeguard rule managed to decrease the average number of function evaluations. The decrease is over 13% for NG algorithm. In that case, the algorithm tried to decrease the sample size in 34% of iterations on average. However, the decrease was not allowed in 21% of trials. On the other hand, in BFGS algorithms, the signal for decreasing came in 24% of iterations, but the decrease was allowed only for half of them.

References

[1] S. Andradottir, A review of simulation optimization techniques, *Proceedings of the 1998 Winter Simulation Conference, 1998, pp. 151-158.*

[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] A. R. Conn, K. Scheinberg, L. N. Vicente, Introduction to Derivative-Free Optimization, *MPS-SIAM Book Series on Optimization, SIAM, Philadelphia, 2009.*

[6] G. Deng, M. C. Ferris, Variable-Number Sample Path Optimization, *Mathematical Programming, Vol. 117, No. 1-2, 2009, pp. 81-109.*

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

[8] M. C. Fu, Gradient Estimation, *S.G. Henderson and B.L. Nelson (Eds.), Handbook in OR & MS, Vol. 13, 2006, pp. 575-616.*

[9] M. C. Fu, Optimization via simulation: A review, *Annals of Operational Research 53, 1994, pp. 199-247.*

[10] T. Homem-de-Mello, Variable-Sample Methods for Stochastic Optimization, *ACM Transactions on Modeling and Computer Simulation, Vol. 13, Issue 2, 2003, pp. 108-133.*

[11] C. Kao, W. T. Song, S. Chen, A modified Quasi-Newton Method for Optimization in Simulation, *Int. Trans. O.R., Vol.4, No.3, 1997, pp. 223-233.*

[12] K. Marti, Solving Stochastical Structural Optimization Problems by RSM-Based Stochastic Approximation Methods - Gradient Estimation in Case of Intermediate Variables, *Mathematical Methods of Operational Research 46, 1997, pp. 409-434.*

[13] M. Montaz Ali, C. Khompatraporn, Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continous Global Optimization Test Problems, *Journal of Global Optimization, Vol. 31, Issue 4, 2005, pp.635-672 .*

[14] J. J. More, S. M. Wild, Benchmarking derivative-free optimization algorithms *SIAM J. Optim, Vol. 20, No. 1, 2009, pp. 172-191.*

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

[16] E. Polak, J. O. Royset, Eficient sample sizes in stochastic nonlinear programing, *Journal of Computational and Applied Mathematics, Vol. 217, Issue 2, 2008, pp. 301-310.*

[17] R.Y. Rubinstein, A. Shapiro, Discrete Event Systems, *John Wiley & Sons, Chichester, England, 1993.*

[18] A. Shapiro, A. Ruszczynski, Stochastic Programming, *Vol. 10 of Handbooks in Operational Research and Management science. Elsevier, 2003, pp. 353-425.*

[19] J. C. Spall, Introduction to Stochastic Search and Optimization, *Wiley-Interscience serises in discrete mathematics, New Jersey, 2003.*