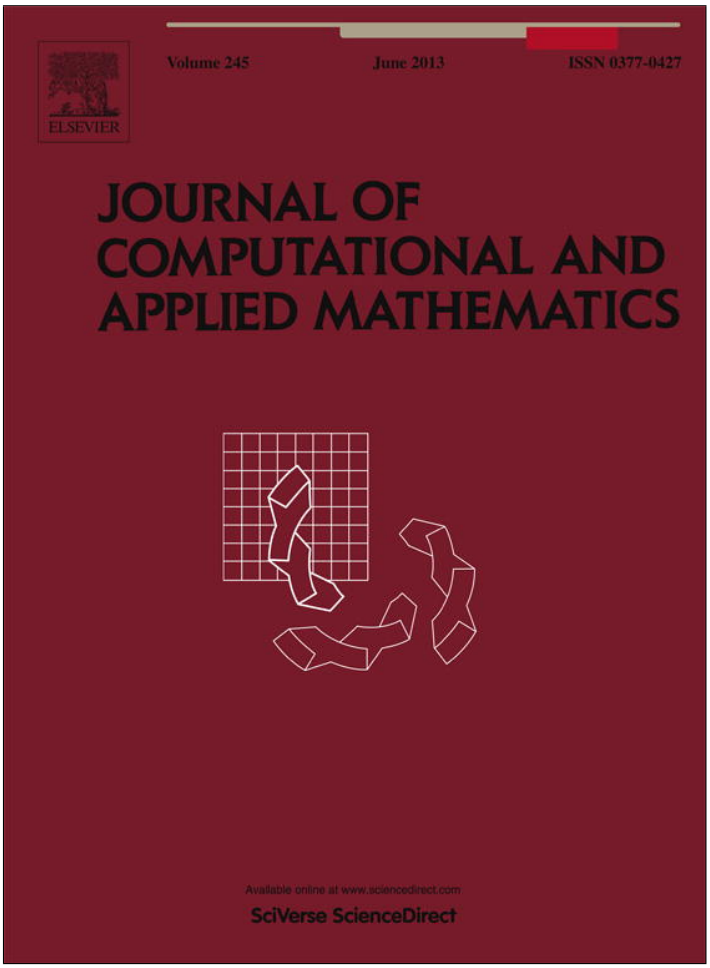

Journal of Computational and Applied Mathematics 245 (2013) 213–231

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Line search methods with variable sample size for unconstrained optimization

Nataša Krejić*, Nataša Krklec

Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia

ARTICLE INFO

Article history:
Received 28 June 2011
Received in revised form 9 October 2012

Keywords:
Stochastic optimization
Line search
Simulations
Sample average approximation
Variable sample size

ABSTRACT

Minimization of unconstrained objective functions in the form of mathematical expectation is considered. The Sample Average Approximation (SAA) method transforms the expectation objective function into a real-valued deterministic function using a 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 requires a very large number of function evaluations. We 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 a decrease of function value are calculated at each iteration. Based on these two measures a new sample size is determined. The rule we present allows us to increase or decrease the sample size at 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 that the variable sample size strategy generates a solution of the same quality as the SAA method but with a significantly smaller number of function evaluations. The algorithm is tested on a couple of examples, including the discrete choice problem.

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 cannot find its analytical form. This is the case even if the analytical form of F is known, which is assumed in this paper. Furthermore we assume that F is bounded with respect to both x and ξ and thus (2) and the mathematical expectation are well defined.

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

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

* Corresponding author. Tel.: +38 1214852864; fax: +38 121350457.
E-mail addresses: natasak@uns.ac.rs, natasa@dmi.uns.ac.rs (N. Krejić), natasa.krklec@dmi.uns.ac.rs (N. Krklec).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2012.12.020

This is the approach that we use as well. Here N represents the size of sample that is used to make approximation (2). An 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 the Law of Large Numbers [1] implies that for every x almost surely

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

In practical applications one cannot have an 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 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 the 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 the subject of many research efforts, see for example [1,2]. The main disadvantage of the SAA method is the need to calculate the expensive objective function defined by (2) at each iteration. 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 the optimization process. One can distinguish two types of variable sample size results. The first type deals with unbounded samples and seeks almost sure convergence. The strategies of this type in general start with small samples and increase their size during the iterative procedure. 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 [3]. The proposed method uses a Bayesian scheme to determine a suitable sample size at 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 [4,5] as well. In [4], convergence is ensured if (3) is satisfied and the sample size possess sufficient growth. The method in [5] states an auxiliary problem that is solved before the optimization process is started and the solution of that auxiliary problem provides an efficient increasing variable sample size strategy.

A refinement of SAA methods based on a careful analysis of the sequence of sample sizes and error tolerance is presented in [6,7]. A set of conditions that ensures almost sure convergence is presented in [7] together with a specific recommendation for sample size and error tolerance sequences. Another interesting approach that offers a quantitative measure of SAA solutions is presented in [8]. Optimality functions for general stochastic programs (expected value objective and constraint functions) are considered. An algorithm that utilizes optimality functions to select the sample size is developed.

The second type of algorithm deals directly with problems of type (4) and seeks convergence towards stationary points of that problem. The algorithms proposed in [9,10] introduce a variable sample size strategy that allows a decrease of the sample size as well as an increase during the optimization process. Roughly speaking, the main idea is to use the decrease of the function value and a measure of the width of the confidence interval to determine the change in sample size. The optimization process is conducted in the trust region framework. We adopt these ideas to the line search framework in this paper and propose an algorithm that allows both an 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 always assume the number of function evaluations [11].

The paper is organized as follows. In Section 2 we state the problem in more detail 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 and in the Appendix.

2. Preliminaries

In order to solve (4) we 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 deterministic [12]. This approach simplifies the definition of stationary points, which is much more complicated in a stochastic environment. It also provides us with standard optimization tools. Various optimization algorithms are described in [13], for example. The one that we 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. The goal is to show how the variable sample scheme presented in [9,10] for trust region methods can be implemented in the line search framework.

Suppose that we are at the iterate 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

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

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

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

for some $\eta \in (0, 1)$. More precisely, starting from $\alpha = 1$, we decrease α by multiplying it with $\beta \in (0, 1)$ until the Armijo condition (6) is satisfied. This can be done in a finite number of trials if the iterate x_k is not a stationary point of $\hat{f}_{N_k}$, assuming that this function is continuously differentiable and bounded from below. For more information about this technique see [13], for example.

After the suitable step size α_k is found, we define the next iterate 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 three parameters: 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 [10,9] and adopted to suit the line search methods while the third parameter is introduced to avoid an unproductive decrease of the sample size as 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 at the current iterate x_k, 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 is an input parameter of our algorithm. We know that $c = \sigma(x_k)\alpha_\delta/\sqrt{N_k}$, where $\sigma(x_k)$ is the standard deviation of 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 cannot find $\sigma(x_k)$, so we use the centered sample variance estimator

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

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 is described in more detail 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 at 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 the 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 at the next iteration. In that case,

we calculate the safeguard parameter ρ_k. It is defined as the 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 an 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 do not allow a decrease of the sample size.

Now, we present the assumptions needed for proving the convergence result of the algorithm that is 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 $\hat{f}_N$, while A3 and A4 allow us to prove the convergence results.

An important consequence of the previous assumptions is that the interchange between the mathematical expectation and the gradient operator is allowed (see [1]), 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, guarantees that the line search is well defined.

Lemma 2.4 (*[13]*)**.** *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 Wolfe conditions* (11) and (12)

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

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

The backtracking technique that we use in order to find a step size that satisfies the Armijo condition (11) generates 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 the 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 the details of such a process.

As already stated, the algorithm is constructed to let the sample size vary across the iterations and to let it decrease if appropriate. Let us state the main algorithm here, leaving the additional ones to be stated later.

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.
If $\|\nabla \hat{f}_{N_k}(x_k)\| > 0$ go to step S4.

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

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

S6 Put $s_k = \alpha_k p_k$, $x_{k+1} = x_k + s_k$ 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 the 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 on 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 the algorithm. The role of the lower sample bound $N_k^{\min}$ is explained after the statement of 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 guarantees 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 for the next iteration. Algorithms 2 and 3 stated below provide details. As already mentioned Algorithm 2 is adopted from [9,10] 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. The update rule for $N_k^{\min}$ is given after Algorithm 3. 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 the algorithm described below. Notice that the following algorithm is constructed to provide $N_k^{\min} \le N_k^+ \le N_{\max}$.

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) $dm_k = \varepsilon_\delta^{N_k}(x_k) \to N_k^+ = N_k$.

(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 $\varepsilon_\delta^{N}(x_k) \to N_k^+$.

(3) $dm_k < \varepsilon_\delta^{N_k}(x_k)$
(i) $dm_k \ge \nu_1 \varepsilon_\delta^{N_k}(x_k)$.
Starting with $N = N_k$, while $dm_k < \varepsilon_\delta^{N}(x_k)$ and $N < N_{\max}$, increase N by 1 and calculate $\varepsilon_\delta^{N}(x_k) \to N_k^+$.
(ii) $dm_k < \nu_1 \varepsilon_\delta^{N_k}(x_k) \to N_k^+ = N_{\max}$.

The basic idea for this kind of reasoning can be found in [9,10]. 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 a sample size as small as possible, if $dm_k > \varepsilon_\delta^{N_k}(x_k)$ we deduce that it is not necessary to have that much precision and we try 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.

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

Let us 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
  - N_{k+1} is a sample size which has not been used so far then $N_{k+1}^{\min} = N_k^{\min}$.
  - N_{k+1} is a sample size which had been used and we have made a big enough decrease of the function $\hat{f}_{N_{k+1}}$ since the last time it has been used, then $N_{k+1}^{\min} = N_k^{\min}$.
  - N_{k+1} is a sample size which had been used and we have not made a big enough decrease of the function $\hat{f}_{N_{k+1}}$ since the last time, 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 $(N_0, \ldots, N_8) = (3, 6, 6, 4, \mathbf{6}, 6, 3, 3, 6)$, 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 $N_{\max}$ is reached and kept until the end.

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 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 cannot be stacked at a size that is lower than the maximal one.

Suppose that there exists $\tilde{n} > n_1$ such that for every $k \geq \tilde{n}$ $N_k = N^1 < N_{\max}$. 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 $s \in \mathbb{N}$

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

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

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

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 cannot 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 first consider the second scenario L2. Notice that this is the case where $N_k^{\min}$ cannot 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

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

Notice that, according to the proposed mechanism for updating $N_k^{\min}$, updating form $N_{k+1}^{\min} = N_{k+1}$ happens only if N_{k+1} is the sample size which had been used already, $N_{k+1} > N_k$ and the obtained decrease in $\hat{f}_{N_{k+1}}$ was not good enough. Therefore, we conclude that there exists an iteration r_1 such that for every $k \geq r_1$ we have one of the following scenarios:

M1 $N_{k+1} \leq N_k$
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 have not used the sample size N_{k+1} before.

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 $\bar{N}$. The definition of $\bar{N}$ implies that there exists iteration r_2 such that for every $k \in \bar{K}_0$, $k \geq r_2$ the sample size is increased to $\bar{N}$, i.e.

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

Define $r = \max\{r_1, r_2\}$ and set $\bar{K} = \bar{K}_0 \bigcap \{r, r+1, \ldots\}$. Clearly, each iteration in $\bar{K}$ excludes the scenario M1. Moreover, taking out the first member of a sequence $\bar{K}$ and retaining the same notation for the remaining sequence we can exclude the scenario M3 as well. This leaves us with M2 as 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, the 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 $k \geq \tilde{n}$, $N_k = N_{\max}$. □

Now, we prove the main result, for which we need to state one more assumption about the search direction.

A5 The sequence of directions p_k generated at S4 of Algorithm 1 is bounded and 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 allows us to consider the general descent direction, but it is obviously satisfied for $p_k = -\nabla \hat{f}_{N_k}(x_k)$. Furthermore quasi-Newton directions also satisfy the assumption under the standard conditions for such methods such as uniform boundedness of the inverse Hessian approximation.

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 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 $k \geq \hat{n} N_k = N_{\max}$ 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 $s \in \mathbb{N}$

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

Again, this inequality and Lemma 2.3 imply

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

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

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

This inequality implies

$$\lim_{k\to\infty}\alpha_k(\nabla\hat{f}_{N_{\max}}(x_k))^T p_k = 0. \tag{16}$$

Now, let x^* be an arbitrary accumulation point of the sequence $\{x_k\}_{k\in\mathbb{N}}$, i.e. let K be the subset $K \subseteq \mathbb{N}$ such that

$$\lim_{k\in K} x_k = x^*.$$

If the sequence of step sizes $\{\alpha_k\}_{k\in K}$ is bounded from below, i.e. if there exists $\hat{\alpha} > 0$ such that $\alpha_k \ge \hat{\alpha}$ for every $k \in K$ sufficiently large, then (16) implies

$$\lim_{k\in K}(\nabla\hat{f}_{N_{\max}}(x_k))^T p_k = 0.$$

This result, together with assumption A5 and Lemma 2.2, implies

$$\nabla\hat{f}_{N_{\max}}(x^*) = \lim_{k\in K}\nabla\hat{f}_{N_{\max}}(x_k) = 0.$$

Now, suppose that there exists a subset $K_1 \subseteq K$ such that $\lim_{k\in K_1}\alpha_k = 0$. This implies the existence of $\hat{k}$ such that for every $k \in K_2 = K_1 \cap \{\max\{\hat{n}, \hat{k}\}, \max\{\hat{n}, \hat{k}\}+1, \ldots\}$ 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

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

By Mean Value Theorem there exists $t_k \in [0, 1]$ such that the previous inequality is equivalent to

$$p_k^T\nabla\hat{f}_{N_{\max}}(x_k + t_k\alpha_k' p_k) > \eta(\nabla\hat{f}_{N_{\max}}(x_k))^T p_k. \tag{17}$$

Notice that $\lim_{k\in K_2}\alpha\prime_k = 0$ and recall that the sequence of search directions is assumed to be bounded. Therefore, there exists p^* and subset $K_3 \subseteq K_2$ such that $\lim_{k\in K_3} p_k = p^*$. Now, taking limit in (17) and using Lemma 2.2, we obtain

$$(\nabla\hat{f}_{N_{\max}}(x^*))^T p^* \ge \eta(\nabla\hat{f}_{N_{\max}}(x^*))^T p^*. \tag{18}$$

On the other hand, we know that $\eta \in (0, 1)$ and p_k is the decreasing direction, i.e. $(\nabla\hat{f}_{N_{\max}}(x_k))^T p_k < 0$ for every $k \in K_3$. This implies that

$$(\nabla\hat{f}_{N_{\max}}(x^*))^T p^* \le 0.$$

The previous inequality and (18) imply that

$$\lim_{k\in K_3}(\nabla\hat{f}_{N_{\max}}(x_k))^T p_k = (\nabla\hat{f}_{N_{\max}}(x^*))^T p^* = 0.$$

Finally, according to assumption A5,

$$\nabla\hat{f}_{N_{\max}}(x^*) = \lim_{k\in K_3}\nabla\hat{f}_{N_{\max}}(x_k) = 0,$$

which completes the proof. □

5. Numerical implementation

In this section, we present some numerical results obtained by Algorithm 1 and compare them with the results obtained by two other methods. The first subsection contains the results obtained on a set of academic test examples, while the second subsection deals with the discrete choice problem, which is relevant in many applications. The test examples presented in 5.1 consist of two different sets. The first one includes Aluffi-Pentini's [14] and Rosenbrock problem [3] in noisy environments. 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. The second set of examples consists of five larger dimension problems in noisy environments taken from [14].

The Mixed Logit problem is slightly different than the problem (4). Given the practical importance of this problem we introduce some minor adjustments of Algorithm 1 and report the results in 5.2. This problem is solved by all considered methods.

All problems are solved by four different implementations of Algorithm 1, two different heuristic methods as suggested by one of the referees and two different implementations of SAA. Let us first describe the exit criteria and cost measurement for all methods and then state the details of their implementation.

As common in numerical testing of noisy problems we are measuring the cost by the number of function evaluations needed for achieving the exit criteria. In all presented examples we stopped when

$$\|\nabla \hat{f}_{N_{\max}}(x_k)\| < 10^{-2}. \tag{19}$$

As the exit criteria implies that the approximate solutions obtained by all methods are of the same quality, the number of function evaluations is a relevant measure of the cost for each method.

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

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

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 with $H_0 = I$, where I denotes the identity matrix. Other possibilities for the initial approximation H_0 can be seen in [13,2]. The inverse Hessian approximation is updated by the BFGS formula that can be found in [13]. More precisely, taking s_k from step S6 of Algorithm 1 and computing

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

after step S8 of Algorithm 1, we obtain

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

The condition $y_k^T s_k > 0$ ensures positive definiteness of the next BFGS update. We enforced this condition or otherwise take $H_{k+1} = H_k$. This way the approximation matrix remains positive definite and provides 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. The BFGS matrix in a noisy environment is analyzed in [15].

If we choose to apply the safeguard rule presented in Algorithm 3, we set the input parameter η_0 to be some finite number. 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 tested here. As all considered methods are implemented with both descent directions, NG and BFGS are used to denote the negative gradient search direction and BFGS search direction in general. The implementations of Algorithm 1 that do not use the safeguard rule, i.e. with $\eta_0 = -\infty$, are denoted by $\rho = -\infty$, while $\rho = \eta_0$ stands for the implementations that use the safeguard rule with the value η_0. The input parameters for Algorithm 2 and for updating the sample size lower bound are

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

Table 1
Stationary points for Aluffi-Pentini's problem.

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

A couple of words are due about the implementation of Step 3 in Algorithm 1. The condition $\|\nabla \hat{f}_{N_k}(x_k)\| = 0$ is replaced by

$$\|\nabla \hat{f}_{N_k}(x_k)\| \leq \max\{0, 10^{-2} - \tilde{\varepsilon}_{\delta}^{N_k}(x_k)\}.$$

Here $\tilde{\varepsilon}_{\delta}^{N_k}(x_k)$ is the measure of the confidence interval for $\|\nabla f(x_k)\|$ around $\|\nabla \hat{f}_{N_k}(x_k)\|$. As the gradient $\nabla \hat{f}_{N_k}(x_k)$ is already available, this measure for the confidence interval is obtained without additional costs, and it improves overall behavior of the considered methods.

One of the referees suggested a heuristic scheme for the sample size, proposing that the first 10 iterations are carried out with the sample size of $0.1\, N_{\max}$, then another 10 iterations with the sample size $0.2N_{\max}$ and so on. We implemented this scheme for both descent directions as in Algorithm 1—the negative gradient and BFGS direction. The scheme suggested by the referee is slightly adjusted to allow us to compare the results with other methods, i.e. to ensure that we get the approximate solution with the same exit criteria as in all other tested methods. We consider the number of iterations used by the corresponding Algorithm 1 (negative gradient or BFGS) method with $\rho = \eta_0$ as the reference number of iterations, say K. Then we perform $0.1K$ iterations (rounded if necessary) with the sample size $0.1N_{\max}$, another $0.1K$ iterations with the sample size $0.2N_{\max}$ and so on until (19) is reached. This way we ensured that the solutions obtained by this scheme are comparable with those obtained by other methods.

Sample Average Approximation Methods are a wide known class of methods that generate a sample of $N_{\max}$ realizations at the beginning of process and proceed with a suitable numerical procedure for solving (4). We tested SAA methods here with both negative gradient and BFGS direction. The same method was also tested within the framework proposed in [6,7], but it did not perform very well and these results are not reported in this paper. Our conjecture is the following: we are dealing with relatively modest $N_{\max}$, so the advantages of the variable scheme from [6,7] could not be seen, as this scheme is intended for $N_{\max} \to \infty$ case.

The line search used for all of the above-described methods is the one defined in Step 5 of Algorithm 1 with the value for the Armijo parameter $\eta = 10^{-4}$. The backtracking is performed with $\beta = 0.5$.

5.1. Numerical results for noisy problems

First of all, we present the numerical results obtained for Aluffi-Pentini's problem, which can be found in [14]. Originally, this is a deterministic problem with box constraints. Following the ideas from [3], some noise is added to the first component of the decision variable and the constraints are removed, 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

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

This problem is solved with three different levels of variance. As we are able to calculate the real objective function and 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.

We conducted 50 independent runs of each algorithm with $x_0 = (1, 1)^T$ and $N_0^{\min} = 3$. 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 (see Table 2). Columns $\|\nabla \hat{f}_{N_{\max}}\|$ and $\|\nabla f\|$ give, respectively, the average values of the gradient norm for the approximate problem (4) and for problem (1), while *fev* represents the average number of function evaluations, with one gradient evaluation being counted as n function evaluations. The last column is added to facilitate comparison and represents the percentage increase/decrease in the number of function evaluations for different methods, with $\rho = 0.7$ being the benchmark method. So, if the number of function evaluations is E_ρ for the benchmark method and E_i is the number of function evaluations for any other method, then the reported number is $(E_i - E_\rho)/E_\rho$.

The methods generated by Algorithm 1 clearly outperform the straightforward SAA method as expected. The heuristic approach is fairly competitive in this example, in particular for problems with smaller variance. The safeguard rule with $\eta_0 = 0.7$ is beneficial in all cases, except for the BFGS direction and $\sigma = 0.1$, where it does not make significant difference in comparison to $\rho = -\infty$. The decrease in the sample size is proposed in approximately 20% of iterations and the safeguard rule is active in approximately half of these iterations.

Table 2
Aluffi-Pentini's problem.

Algorithm	NG				BFGS			
	$\|\nabla \hat{f}_{N_{\max}}\|$	$\|\nabla f\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	$\|\nabla f\|$	*fev*	%
$\sigma^2 = 0.01$, $N_{\max} = 100$								
$\rho = -\infty$	0.00747	0.01501	1 308	9.01	0.00389	0.01208	811	6.64
$\rho = 0.7$	0.00767	0.01496	1 200	0.00	0.00365	0.01279	761	0.00
Heur	0.00618	0.01480	1 250	4.24	0.00407	0.01383	852	12.04
SAA	0.00844	0.01378	1 832	52.73	0.00527	0.01398	940	23.55
$\sigma^2 = 0.1$, $N_{\max} = 200$								
$\rho = -\infty$	0.00722	0.03499	3 452	7.84	0.00363	0.03530	1 948	−0.38
$\rho = 0.7$	0.00718	0.03435	3 201	0.00	0.00341	0.03466	1 955	0.00
Heur	0.00658	0.03531	3 556	11.09	0.00414	0.03460	2 284	16.81
SAA	0.00793	0.03005	4 264	33.23	0.00392	0.03051	2 928	49.75
$\sigma^2 = 1$, $N_{\max} = 600$								
$\rho = -\infty$	0.00540	0.06061	13 401	17.78	0.00303	0.06110	8 478	15.53
$\rho = 0.7$	0.00528	0.06071	11 378	0.00	0.00358	0.06116	7 338	0.00
Heur	0.00492	0.05843	13 775	21.07	0.00344	0.05656	8 719	18.81
SAA	0.00593	0.05734	15 852	39.32	0.00336	0.06444	14 784	101.46

Table 3
The approximate stationary points for Aluffi-Pentini's problem.

Algorithm	NG					BFGS				
	g	*l*	max	*fgm*	*flm*	*g*	*l*	max	*fgm*	*flm*
$\sigma^2 = 0.01$, $N_{\max} = 100$										
$\rho = -\infty$	0	50	0	–	−0.14524	0	50	0	–	−0.14543
$\rho = 0.7$	0	50	0	–	−0.14542	0	50	0	–	−0.14543
Heur	0	50	0	–	−0.14542	0	50	0	–	−0.14543
SAA	0	50	0	–	−0.14542	0	50	0	–	−0.14543
$\sigma^2 = 0.1$, $N_{\max} = 200$										
$\rho = -\infty$	14	35	1	−0.11712	−0.12887	14	36	0	−0.11710	−0.12818
$\rho = 0.7$	17	32	1	−0.11507	−0.13104	14	36	0	−0.11710	−0.12818
Heur	20	30	0	−0.11364	−0.13275	15	35	0	−0.11635	−0.12882
SAA	1	49	0	−0.10523	−0.12551	1	49	0	−0.10533	−0.12548
$\sigma^2 = 1$, $N_{\max} = 600$										
$\rho = -\infty$	35	15	0	−0.12674	−0.097026	37	13	0	−0.12047	−0.13036
$\rho = 0.7$	36	14	0	−0.11956	−0.11337	36	14	0	−0.11982	−0.13133
Heur	34	16	0	−0.12114	−0.11079	28	22	0	−0.11887	−0.12835
SAA	33	17	0	−0.11745	−0.11857	50	0	0	−0.11230	–

Given that the considered problems have more than one stationary point we report the distribution of the approximate stationary points in Table 3. Columns *global*, *local* and *max* count the numbers of replicants converging to the global minimizer, local minimizer and maximizer respectively. Columns *fgm* and *flm* represent the average values of function f in the runs that converged to the global minimizer and local minimizer, respectively.

All methods behave more or less similarly. Notice that as the variance increases, the number of replications that are converging towards the global minimizers increases as well. However, we also registered convergence towards maximizers when the variance is increased. The only exception from this relatively similar behavior of all methods appears to happen for $\sigma = 0.1$, where SAA strongly favors the local minimizers while all other methods converge to the global minimizers more frequently.

The next example is based on the Rosenbrock function. Following the example from [3], the noise is added to the first component in order to make it random. The following objective function is thus obtained

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

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 the 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.

Table 4
Rosenbrock problem—the global minimizers.

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 5
Rosenbrock problem.

BFGS				
Algorithm	$\|\nabla \hat{f}_{N_{\max}}\|$	$\|\nabla f\|$	*fev*	%
$\sigma^2 = 0.001$, $N_{\max} = 3500$				
$\rho = -\infty$	0.003939	0.208515	44 445	7.51
$\rho = 0.7$	0.003595	0.208355	41 338	0.00
Heur	0.002521	0.206415	127 980	209.59
SAA	0.003241	0.208450	247 625	499.03
$\sigma^2 = 0.01$, $N_{\max} = 3500$				
$\rho = -\infty$	0.003064	0.132830	58 944	7.74
$\rho = 0.7$	0.003170	0.132185	54 711	0.00
Heur	0.001968	0.132730	114 070	108.5
SAA	0.003156	0.132155	216 825	296.3
$\sigma^2 = 0.1$, $N_{\max} = 3500$				
$\rho = -\infty$	0.003387	0.091843	70 958	3.49
$\rho = 0.7$	0.003359	0.091778	68 566	0.00
Heur	0.002259	0.091167	106 031	54.64
SAA	0.003279	0.092130	161 525	135.58

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 a 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 approximation is set to be $x_0 = (-1, 1.2)^T$.

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, a clear advantage is assigned to the algorithm that uses the safeguard rule. The heuristic sample size scheme does not appear to be well suited for this example, although the performance improves significantly as the variance increases. The percentage of iterations where the decrease of a sample size is considered increases with the noise and varies from 7% for $\sigma = 0.001$ to 30% for $\sigma = 1$. The rejection due to the safeguard rule from Algorithm 3 also differs, from 15% in the first case to 33% in the case with the largest variance.

Let us now present the results for larger dimension problems. We consider the set of five problems, each one of the dimension 10. The problems from [14] are stated below together with their initial approximations x_0.

- Exponential problem

$$f(x) = E\left(-e^{-0.5\|\xi x\|^2}\right), \qquad x_0 = (0.5, \ldots, 0.5)^T.$$

- Griewank problem

$$f(x) = E\left(1 + \frac{1}{4000}\|\xi x\|^2 - \prod_{i=1}^{10} \cos\left(\frac{x_i \xi}{\sqrt{i}}\right)\right), \qquad x_0 = (10, \ldots, 10)^T.$$

- Neumaier 3 problem

$$f(x) = E\left(\sum_{i=1}^{10} (\xi x_i - 1)^2 - \sum_{i=2}^{10} \xi^2 x_i x_{i-1}\right), \qquad x_0 = (1, \ldots, 1)^T.$$

- Salomon problem

$$f(x) = E\left(1 - \cos(2\pi \|\xi x\|^2) + 0.1\|\xi x\|^2\right), \qquad x_0 = (2, \ldots, 2)^T.$$

- Sinusoidal problem

$$f(x) = E\left(-A \prod_{i=1}^{10} \sin(\xi x_i - z) - \prod_{i=1}^{10} \sin(B(\xi x_i - z))\right),$$
$$A = 2.5, \qquad B = 5, \qquad z = 30, \qquad x_0 = (1, \ldots, 1)^T.$$

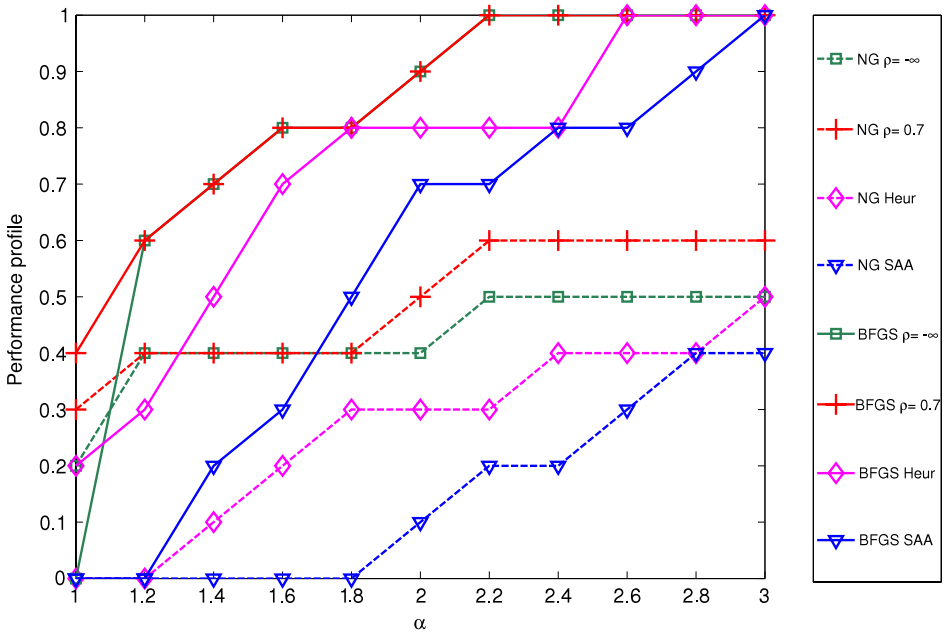

Fig. 1. Performance profile.

The noise component ξ is taken as a Normal random variable $\mathcal{N}(1, \sigma^2)$ for different values of σ as specified in Tables 7–11 in Appendix. All results are obtained taking $N_0^{\min} = 3$ with exit criteria (19). The Armijo parameter is $\eta = 10^{-4}$ while the backtracking is performed with $\beta = 0.5$. The considered methods are again the same—four variants of Algorithm 1 (the negative gradient with $\rho = -\infty$ and $\rho = 0.7$ and the BFGS methods with $\rho = -\infty$ and $\rho = 0.7$), the heuristic sample size scheme and the SAA method, in total 8 methods. Two levels of noise $\sigma^2 = 0.1$ and $\sigma^2 = 1$ are considered for each of the five problems, resulting in the set of 10 problems.

The results are presented using the performance profile [16,17] in Fig. 1. As the performance profile clearly indicates, all implementations of Algorithm 1 clearly outperformed both the heuristic and SAA corresponding methods.

A natural question here is the dynamics of the variable sample scheme and the actual influence of the decrease as well as the safeguard rule. The number of iterations where $N_k^+ < N_k$ varies very much through the set of examples and variances. The Griewank test function is solved by both NG methods without any decrease at all. A number of BFGS iterations where $N_k^+ < N_k$ occurred was also rather small and the average number of safeguard rule calls varies from 11% to 20% for this example, and none of the decreases is beneficial in terms of function evaluations. This is the only example where the heuristic scheme is the best method for both directions. In all other examples a decrease in the sample size occurs and the safeguard is applied. However the numbers are rather different, ranging from a couple of percent to almost one half of the iterations. The same range is valid for the rejection of the decrease according to the safeguard rule. The average number of iterations where $N_k^+ < N_k$ for all tested examples and both NG and BFGS methods is 14.87%. The decrease is judged as unproductive and it is rejected in 20.57% of cases on average. It is quite clear that the safeguard rule, i.e. the appropriate value of the parameter which determines the acceptance or rejection of the decrease, is problem dependent. In this paper we report results for the same value of that parameter for all examples and methods to make the comparison fair, as all other parameters have the same values for all problems. But further research is definitely needed to specify some kind of recommendation for the safeguard rule that will guarantee its efficient use, and we plan to investigate that in future work.

To conclude this discussion the plot of the sample scheme dynamic for the Sinusoidal problem and one noise realization with $\sigma = 1$ and NG direction is shown in Fig. 2. The NG $\rho = 0.7$ method requested 26 219 function evaluations, while NG with $\rho = -\infty$ took 40 385 function evaluations, and NG Heur 39 983 function evaluations. As in almost all examples SAA NG is the worst requiring 86 500 function evaluations. One can see in Fig. 2 that the safeguard rule rejects the decrease at the 6th iteration and keeps the maximal sample size until the end of the process, while the method with $\rho = -\infty$ performed a number of sample decreases which are in fact unproductive in this example.

A more detailed account of these tests is available in the Appendix, Tables 7–11. The structure of the tables is the same as before—the columns are the value of the sample gradient at the last iteration, the cost measured as the number of function evaluations and the column showing the relative increase/decrease for different methods. The cost of Algorithm 1 with the safeguard is taken as the benchmark. All algorithms are tested in 20 independent runs and the reported numbers are the average values of these 20 runs. The same sample realizations are used for all methods.

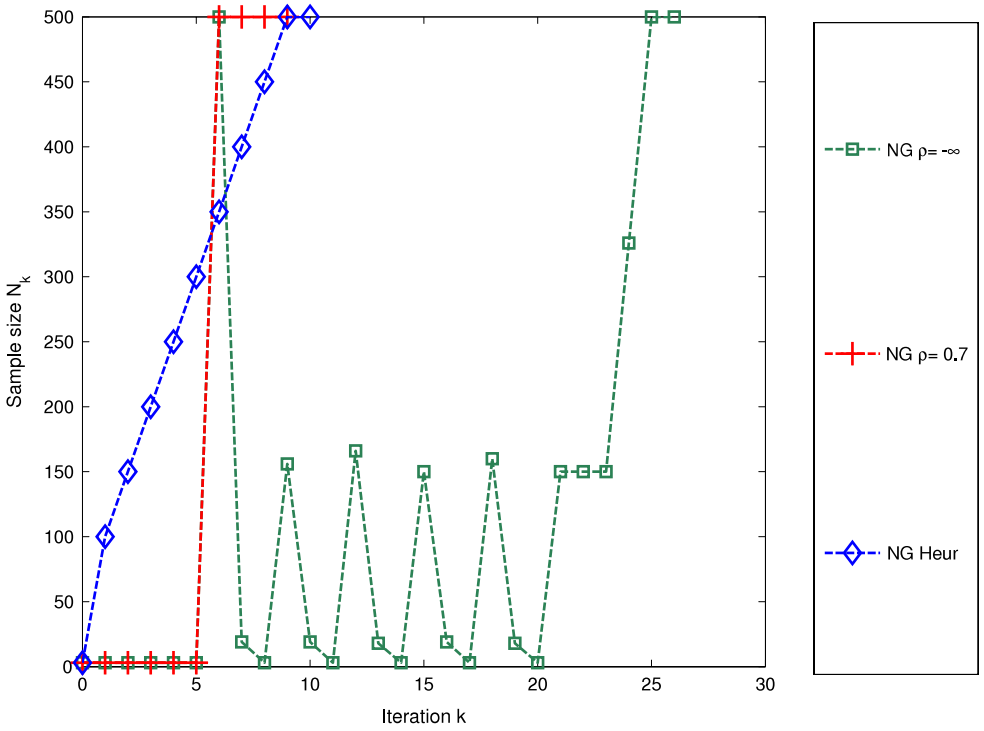

Fig. 2. Sample size versus iteration.

5.2. Application to discrete choice theory—Mixed Logit models

In this section we present numerical results obtained by applying slightly modified algorithms on simulated data. Discrete choice problems are the subject of various disciplines such as econometrics, transportation, psychology etc. The problem that we considered is an unconstrained parameter estimation problem. We briefly describe the problem here, while the more detailed description with further references can be found in [9,10,18].

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 defined by $m_j = (k_1^j, \ldots, k_{r_k}^j)^T$ and $\varepsilon_{i,j}$ is the error term. We observe probably the most popular model in practice where $V_{i,j}$ is a linear function, that is

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

The vector $\beta^i,\ i = 1, 2, \ldots, r_a$ has r_k components, all of them normally distributed. More precisely,

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

where $\xi_j^i,\ i = 1, 2, \ldots, r_a,\ j = 1, 2, \ldots, r_k$ are i.i.d. random variables with a standardized Normal distribution. In other words, $\beta_k^i : \mathcal{N}(\mu_k, \sigma_k^2)$ for every i. The parameters μ_k and $\sigma_k,\ k = 1, 2, \ldots, r_k$ are the ones that should be estimated. Therefore, the vector of unknowns is

$$x = (\mu_1, \ldots, \mu_{r_k}, \sigma_1, \ldots, \sigma_{r_k})^T$$

and the dimension of our problem is $n = 2r_k$. Thus $V_{i,j}$ is a function of x and the random vector ξ^i,

$$V_{i,j} = m_j^T \beta^i(x, \xi^i) = \sum_{s=1}^{r_k} k_s^j (x_s + \xi_s^i x_{r_k+s}) = V_{i,j}(x, \xi^i).$$

The term $\varepsilon_{i,j}$ is a random variable whose role is to collect all factors which 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 assume that for every i and every j the random variable $\varepsilon_{i,j}$ follows the Gumbel distribution with mean 0 and scale parameter 1. The Gumbel distribution is also known as the type 1 extreme value distribution.

Now, suppose that every agent makes his own choice among these alternatives. The problem is to maximize the likelihood function. Under the assumptions that are stated above, if the realization $\bar{\xi}^i$ of $\xi^i = (\xi^i_1, \ldots, \xi^i_{r_k})^T$ 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 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

$$\max_{x\in\mathbb{R}^n} \prod_{i=1}^{r_a} P_{i,j(i)}(x). \tag{24}$$

The equivalent form of (24) 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 is

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

Here $\xi^i_1, \ldots, \xi^i_N$ are independent realizations of the random vector ξ^i. The realizations are independent across the agents as well. Calculating the exact gradient of $\hat{f}_N$ is affordable and the 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, $\varepsilon^N_\delta(x)$. We define the approximation of the confidence interval radius as it is proposed in [18],

$$\varepsilon^N_\delta(x) = \frac{\alpha_\delta}{r_a} \sqrt{\sum_{i=1}^{r_a} \frac{\hat{\sigma}^2_{N,i,j(i)}(x)}{N P^2_{i,j(i)}(x)}}. \tag{25}$$

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.

$$\hat{\sigma}^2_{N,i,j(i)}(x) = \frac{1}{N-1} \sum_{s=1}^{N} \left(L_{i,j(i)}(x, \xi^i_s) - \frac{1}{N} \sum_{k=1}^{N} L_{i,j(i)}(x, \xi^i_k)\right)^2.$$

The confidence level that is used for numerical testing is retained at 0.95, therefore $\alpha_\delta \approx 1.96$. The reason for taking (25) is the fact that it can be shown, by using the Delta method [19,1], that $\sqrt{N}(f(x) - \hat{f}_N(x))$ converges in distribution towards the 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–A4 can be reformulated in the following way

B2 For every N, $\hat{f}_N \in C^1(\mathbb{R}^n)$.
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 a positive constant M_{FF} such that for every N, x, $\hat{f}_N(x) \leq M_{FF}$.

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* $\varepsilon^{N_k}_\delta(x_k) \geq \kappa$ *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* $\hat{f}_{N_{\max}}$.

Table 6
Mixed Logit problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\rho = -\infty$	0.00887	3.98E+07	−8.50	0.00550	5.65E+07	25.16
$\rho = 0.7$	0.00896	4.36E+07	0.00	0.00523	4.52E+06	0.00
Heur	0.00842	1.09E+08	151.68	0.00674	1.53E+07	237.94
SAA	0.00929	8.07E+07	85.41	0.00810	1.82E+07	303.98

Table 7
Exponential problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\sigma^2 = 0.1, N_{\max} = 200$						
$\rho = -\infty$	0.00087	4 591	0.00	0.00154	4 604	0.00
$\rho = 0.7$	0.00087	4 591	0.00	0.00154	4 604	0.00
Heur	0.00111	7 033	53.18	0.00131	7 018	52.42
SAA	0.00314	11 600	152.64	0.00081	12 200	164.97
$\sigma^2 = 1, N_{\max} = 500$						
$\rho = -\infty$	0.00313	47 454	68.71	0.00088	15 752	2.53
$\rho = 0.7$	0.00217	28 128	0.00	0.00138	15 364	0.00
Heur	0.00400	575 270	1945.22	0.00054	21 668	41.04
SAA	0.00474	668 025	2274.99	0.00268	36 250	135.95

The test problem is generated as follows. We consider five alternatives with five characteristics for each alternative. Thus we generate the matrix M from $\mathbb{R}^{5\times 5}$ using the standardized Normal distribution such that each column of M represents the characteristics of one of the alternatives. The number of agents is assumed to be 500. So the matrix $B \in \mathbb{R}^{5\times 500}$ is generated with $B_{ij} : \mathcal{N}(0.5, 1)$ and each column of that matrix represents one realization of the random vector β^i. Finally, the matrix of random terms ε_{ij} from $\mathbb{R}^{5\times 500}$ is formed such that each component is a realization of a random variable with the Gumbel distribution with parameters 0 and 1. These three matrices are used to find the vector of choices for 500 agents.

The results presented in Table 6 are obtained after 10 independent runs of each algorithm. At each run, the initial approximation 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 250 000. 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.

According to *fev* columns, the algorithms with variable sample size strategy once again perform better than their fixed-size counterparts. The heuristic method does not perform well in this example. Notice also that the safeguard rule implies the decrease of the average number of function evaluations significantly in the case of the BFGS method, but it produces a relatively small negative effect for the NG method, increasing the number of function evaluations. In the case of the BFGS method the decrease in the sample size is implied in 16.67% of iterations, but the safeguard rule declines the decrease in 58.33% of these iterations. For the NG method the corresponding numbers are 26.84% and 20.88%.

Several values for the safeguard are tested and the results differ depending on that value. It is rather natural to conjecture that each problem requires its own "optimal" safeguard parameter, and this question will be the subject of further research. However we report the results obtained with 0.7 here to maintain the consistency of all presented numerical results in this paper.

Acknowledgments

We are grateful to the anonymous referees whose constructive remarks helped us to improve this paper. The first author's research was supported by Serbian Ministry of Education and Science, grant no. 174030. The second author's research was supported by Serbian Ministry of Education and Science, grant no. 174030.

Appendix

See Tables 7–11.

Table 8
Griewank problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\sigma^2 = 0.1, N_{\max} = 500$						
$\rho = -\infty$	0.00997	1 796 250	0.00	0.00795	312 840	−0.86
$\rho = 0.7$	0.00997	1 796 250	0.00	0.00822	315 550	0.00
Heur	0.00988	1 160 300	−35.40	0.00505	172 490	−45.34
SAA	0.00996	1 800 750	0.25	0.00794	504 425	59.86
$\sigma^2 = 1, N_{\max} = 1000$						
$\rho = -\infty$	0.00993	6 343 500	0.00	0.00758	408 585	1.98
$\rho = 0.7$	0.00993	6 343 500	0.00	0.00759	400 670	0.00
Heur	0.00995	3 790 300	−40.25	0.00537	264 070	−34.09
SAA	0.00994	6 355 500	0.19	0.00698	340 150	−15.10

Table 9
Neumaier 3 problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\sigma^2 = 0.1, N_{\max} = 500$						
$\rho = -\infty$	0.00732	798 625	−1.85	0.00305	30 223	0.80
$\rho = 0.7$	0.00714	813 685	0.00	0.00306	29 984	0.00
Heur	0.00598	725 680	−10.82	0.00384	40 338	34.53
SAA	0.00663	1 052 025	29.29	0.00278	54 825	82.85
$\sigma^2 = 1, N_{\max} = 2000$						
$\rho = -\infty$	0.00949	3 050 850	0.17	0.00421	138 195	2.71
$\rho = 0.7$	0.00948	3 045 650	0.00	0.00354	134 555	0.00
Heur	0.00945	2 199 650	−27.78	0.00503	161 140	19.76
SAA	0.00937	3 496 200	14.79	0.00128	190 000	41.21

Table 10
Salomon problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\sigma^2 = 0.1, N_{\max} = 500$						
$\rho = -\infty$	0.00411	26 590	8.55	0.00376	30 814	−7.26
$\rho = 0.7$	0.00396	24 495	0.00	0.00297	33 226	0.00
Heur	0.00569	54 620	122.99	0.00243	59 057	77.74
SAA	0.00497	44 750	82.69	0.00452	30 250	−8.96
$\sigma^2 = 1, N_{\max} = 2000$						
$\rho = -\infty$	0.00164	75 078	−16.20	0.00234	154 245	0.00
$\rho = 0.7$	0.00157	89 595	0.00	0.00235	154 245	0.00
Heur	0.00153	127 920	42.78	0.00214	182 650	18.42
SAA	0.00272	196 100	118.87	0.00349	143 100	−7.23

Table 11
Sinusoidal problem.

Algorithm	NG			BFGS		
	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%	$\|\nabla \hat{f}_{N_{\max}}\|$	*fev*	%
$\sigma^2 = 0.1, N_{\max} = 200$						
$\rho = -\infty$	0.00525	22 578	2.99	0.00169	10 518	0.54
$\rho = 0.7$	0.00520	21 923	0.00	0.00125	10 461	0.00
Heur	0.00457	29 512	34.61	0.00202	18 450	76.36
SAA	0.00575	32 860	49.89	0.00326	18 470	76.56
$\sigma^2 = 1, N_{\max} = 500$						
$\rho = -\infty$	0.00473	30 968	2.14	0.00349	30 550	−0.60
$\rho = 0.7$	0.00449	30 320	0.00	0.00339	30 735	0.00
Heur	0.00385	40 453	33.42	0.00338	37 588	22.30
SAA	0.00527	65 475	115.95	0.00473	48 525	57.88

References

[1] A. Shapiro, A. Ruszczynski, Stochastic Programming, in: Handbooks in Operational Research and Management Science, vol. 10, Elsevier, 2003, pp. 353–425.
[2] J.C. Spall, Introduction to Stochastic Search and Optimization, in: Wiley-Interscience Serises in Discrete Mathematics, New Jersey, 2003.
[3] G. Deng, M.C. Ferris, Variable-number sample path optimization, Mathematical Programming 117 (1–2) (2009) 81–109.
[4] T. Homem-de-Mello, Variable-sample methods for stochastic optimization, ACM Transactions on Modeling and Computer Simulation 13 (2) (2003) 108–133.
[5] E. Polak, J.O. Royset, Eficient sample sizes in stochastic nonlinear programing, Journal of Computational and Applied Mathematics 217 (2) (2008) 301–310.
[6] R. Pasupathy, On choosing parameters in restrospective-approximation algorithms for simulation-optimization, in: L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto, (Eds.) Proceedings of the 2006 Winter Simulation Conference, pp. 208–215.
[7] R. Pasupathy, On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization, Operations Research 58 (4) (2010) 889–901.
[8] J.O. Royset, Optimality functions in stochastic programming, Mathematical Programming 135 (1–2) (2012) 293–321.
[9] F. Bastin, Trust-region algorithms for nonlinear stochastic programming and Mixed Logit models, Ph.D. Thesis, University of Namur, Belgium, 2004.
[10] F. Bastin, C. Cirillo, P.L. Toint, An adaptive Monte Carlo algorithm for computing Mixed Logit estimators, Computational Management Science 3 (1) (2006) 55–79.
[11] J.J. More, S.M. Wild, Benchmarking derivative-free optimization algorithms, SIAM Journal on Optimization 20 (1) (2009) 172–191.
[12] M.C. Fu, Optimization via simulation: a review, Annals of Operations Research 53 (1994) 199–247.
[13] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 1999.
[14] 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 31 (4) (2005) 635–672.
[15] C. Kao, W.T. Song, S. Chen, A modified quasi-Newton method for optimization in simulation, International Transactions in Operational Research 4 (3) (1997) 223–233.
[16] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2) (2002) 201–213.
[17] COPS, See http://www.mcs.anl.gov/more/cops/.
[18] F. Bastin, C. Cirillo, P.L. Toint, Convergence theory for nonconvex stochastic programming with an application to Mixed Logit, Mathematical Programming Series B 108 (2006) 207–234.
[19] R.Y. Rubinstein, A. Shapiro, Discrete Event Systems, John Wiley & Sons, Chichester, England, 1993.