Variance Reduction in Population-Based Optimization: Application to Unit Commitment

Jean-Joseph Christophe, Jérémie Decock, Jialin Liu, Olivier Teytaud

Variance Reduction in Population-Based Optimization: Application to Unit Commitment

Jean-Joseph Christophe, Jérémie Decock, Jialin Liu, Olivier Teytaud

Bat 660 Claude Shannon Univ. Paris-Sud, 91190 Gif-sur-Yvette, France
{firstname.lastname}@inria.fr
https://tao.lri.fr

Abstract. We consider noisy optimization and some traditional variance reduction techniques aimed at improving the convergence rate, namely (i) common random numbers (CRN), which is relevant for population-based noisy optimization and (ii) stratified sampling, which is relevant for most noisy optimization problems. We present artificial models of noise for which common random numbers are very efficient, and artificial models of noise for which common random numbers are detrimental. We then experiment on a desperately expensive unit commitment problem. As expected, stratified sampling is never detrimental. Nonetheless, in practice, common random numbers provided, by far, most of the improvement.

Keywords: Noisy Optimization, Variance Reduction, Stratified Sampling, Common Random Numbers

1 Introduction

1.1 Noisy black-box optimization

We consider a function $f(x, w)$, with x in a d-dimensional search domain and w a random variable with values in $D \subset \mathbb{R}$. We assume that the optimization algorithm has only access to independent random realizations of $f(x, w)$. The goal of the optimization algorithm is to approximate $x^* = \arg\min_{x \in \mathbb{R}^d} \mathbb{E}_w[f(x, w)]$.

1.2 Noisy optimization with variance reduction

In standard noisy optimization frameworks, the black-box noisy optimization algorithm, for its n^{th} request to the black-box objective function, can only provide some x and receive a realization of $f(x, w_n)$. The w_n, $n \in \{1, 2, \dots\}$, are independent samples of w. The algorithm can not influence the w_n. Contrarily to this standard setting, we here assume that the algorithm can request $f(x, w_n)$ where w_n is:

- either an independent copy of w (independent of all previously used values);

– or a previously used value w_m for some $m < n$ (m is chosen by the optimization algorithm).

Due to this possibility, *paired sampling* can be applied, i.e. the same w_n can be used several times, as explained later. In addition, we assume that we have *strata*. A stratum is a subset of D. Strata have empty intersections and their union is D (i.e. they realize a partition of D). When an independent copy of w is requested, the algorithm can decide to provide it conditionally to a chosen stratum. Thanks to strata, we can apply *stratified sampling* (Section 1.3).

1.3 Statistics of variance reduction

Monte Carlo methods are the estimation of the expected value of a random variable owing to a randomly drawn sample. Typically, in our context, $\mathbb{E}[f(x,w)]$ can be estimated as a result of $f(x,w_1), f(x,w_2), \ldots, f(x,w_n)$, where the w_i are independent copies of w, $i \in \{1,\ldots,n\}$. Laws of large numbers prove, under various assumptions, the convergence of Monte Carlo estimates such as (see [2])

$$\hat{\mathbb{E}}f(x,w) = \frac{1}{n}\sum_{i=1}^{n} f(x,w_i) \to \mathbb{E}_w f(x,w). \tag{1}$$

There are also classical techniques for improving the convergence:

– *Antithetic variates* (symmetries): ensure some regularity of the sampling by using symmetries. For example, if the random variable w has distribution invariant by symmetry w.r.t 0, then, instead of Eq. 1, we use Eq. 2, which reduces the variance:

$$\hat{\mathbb{E}}f(x,w)=\frac{1}{n}\sum_{i=1}^{n/2}\left(f(x,w_i)+f(x,-w_i)\right). \tag{2}$$

  More sophisticated antithetic variables are possible (combining several symmetries).
– *Importance sampling*: instead of sampling w with density dP, we sample w' with density dP'. We choose w' such that the density dP' of w' is higher in parts of the domain which are critical for the estimation. However, this change of distribution introduces a bias. Therefore, when computing the average, we change the weights of individuals by the ratio of probability densities as shown in Eq. 3 - which is an unbiased estimate.

$$\hat{\mathbb{E}}f(x,w)=\frac{1}{n}\sum_{i=1}^{n}\frac{dP(w_i)}{dP'(w_i)}f(x,w_i) \tag{3}$$

– *Quasi Monte Carlo* methods: use samples aimed at being as uniform as possible over the domain. Quasi Monte Carlo methods are widely used in integration; thanks to modern randomized Quasi Monte Carlo methods, they

are usually at least as efficient as Monte Carlo and much better in favorable situations [16, 3, 13, 24]. There are interesting (but difficult and rather "white-box") tricks for making them applicable for time-dependent random processes with many time steps [15].

- [6] proposes to generate a finite sample which approximates a random process, optimally for some metric. This method has advantages when applied in the framework of Bellman algorithms as it can provide a tree representation, mitigating the anticipativity issue. But it is hardly applicable when aiming at the convergence to the solution for the underlying random process.
- *Control variates*: instead of estimating $\mathbb{E}f(x,w)$, we estimate $\mathbb{E}\left(f(x,w)-g(x,w)\right)$, using

$$\mathbb{E}f(x,w) = \underbrace{\mathbb{E}g(x,w)}_{A} + \mathbb{E}\underbrace{\left(f(x,w)-g(x,w)\right)}_{B}.$$

  This makes sense if g is a reasonable approximation of f (so that term B has a small variance) and term A can be computed quickly (e.g. if computing g is much faster than computing f or A can be computed analytically).
- *Stratified sampling* is the case in which each w_i is randomly drawn conditionally to a stratum. We consider that the domain of w is partitioned into disjoint strata $S_1, \ldots, S_N$. N is the number of strata. The stratification function $i \mapsto s(i)$ is chosen by the algorithm and w_i is randomly drawn conditionally to $w_i \in S_{s(i)}$.

$$\hat{\mathbb{E}}f(x,w) = \sum_{i=1}^{n} \frac{P(w \in S_{s(i)}) f(x,w_i)}{\text{Cardinality}\{j \in \{1,\ldots,n\}; w_j \in S_{s(i)}\}} \tag{4}$$

- *Common random numbers (CRN)*, or paired comparison, refer to the case where we want to know $\mathbb{E}f(x,w)$ for several x, and use the same samples $w_1, \ldots, w_n$ for the different possible values of x.

In this paper, we focus on stratified sampling and paired sampling, in the context of optimization with arbitrary random processes. They are easy to adapt to such a context, which is not true for other methods cited above.

Stratified sampling Stratified sampling involves building strata and sampling in these strata.

Simultaneously building strata and sampling There are some works doing both simultaneously, i.e. build strata adaptively depending on current samples. For example, [11, 18] present an iterative algorithm which stratifies a highly skewed population into a take-all stratum and a number of take-some strata. [10] improves their algorithm by taking into account the gap between the variable used for stratifying and the random value to be integrated.

A priori stratification However, frequently, strata are built in an ad hoc manner depending on the application at hand. For example, an auxiliary variable $\tilde{f}(x^*, w)$ might approximate $w \mapsto f(x^*, w)$, and then strata can be defined as a partition of the $\tilde{f}(x^*, w)$. It is also convenient for visualization, as in many cases the user is interested in viewing statistics for w leading to extreme values of $f(x^*, w)$. More generally, two criteria dictate the choice of strata:

- a small variance inside each stratum, i.e. $\mathcal{V}ar_{w|S} f(x^*, w)$ small for each stratum S, is a good idea;
- interpretable strata for visualization purpose.

The sampling can be

- *proportional*, i.e. the number of samples in each stratum S is proportional to the probability $P(w \in S)$;
- or *optimal*, i.e. the number of samples in each stratum S is proportional to a product of $P(w \in S)$ and an approximation of the standard deviation $\sqrt{\mathcal{V}ar_{w|S} f(x^*, w)}$. In this case, reweighting is necessary, as in Eq. 4.

Stratified noisy optimization Compared to classical stratified Monte Carlo, an additional difficulty when working in stratified noisy optimization is that x^* is unknown, so we can not easily sample $f(x^*, w)$. Also, the strata should be used for many different x; if some of them are very different, nothing guarantees that the variance $Var_{w|S} f(x, w)$ is approximately the same for each x and for x^*. As a consequence, there are few works using stratification for noisy optimization and there is, to the best of our knowledge, no work using optimal sampling for noisy optimization, although there are many works around optimal sampling. We will here focus on the simple proportional case. In some papers[12], the word "stratified" is used for *Latin Hypercube Sampling*; we do not use it in that sense in the present paper.

Common random numbers & paired sampling Common Random Numbers (CRN), also called correlated sampling or pairing, is a simple but powerful technique for variance reduction in noisy optimization problems. Consider $x_1, x_2 \in \mathbb{R}^d$, where d is the dimension of the search domain and w_i denotes the i^{th} independent copy of w:

$$\begin{aligned} & \mathcal{V}ar \sum_{i=1}^{n} (f(x_1, w_i) - f(x_2, w'_i)) \\ &= n\mathcal{V}ar \left(f(x_1, w_1) - f(x_2, w'_1)\right) \\ &= n\mathcal{V}ar f(x_1, w_1) + n\mathcal{V}ar f(x_2, w'_1) \\ &\quad -2n\mathcal{C}ov \left(f(x_1, w_1), f(x_2, w'_1)\right). \end{aligned}$$

If $\mathcal{C}ov(f(x_1, w_i), f(x_2, w'_i)) > 0$, i.e. there is a positive correlation between $f(x_1, w_i)$ and $f(x_2, w'_i)$, the estimation errors are smaller. CRN is based on $w_i = w'_i$, which is usually a simple and efficient solution for correlating $f(x_1, w_i)$

and $f(x_2, w'_i)$; there are examples in which, however, this does not lead to a positive correlation. In Section 2.2, we will present examples in which CRN does not work.

Pairing in artificial intelligence Pairing is used in different application domains related to optimization. In games, it is a common practice to compare algorithms based on their behaviors on a finite constant set of examples [8]. The cost of labelling (i.e. the cost for finding the ground truth regarding the value of a game position) is a classical reason for this. This is different from simulating against paired random realizations (because it is usually an adversarial context rather than a stochastic one), though it is also a form of pairing and is related to our framework of dynamic optimization. More generally, paired statistical tests improve the performance of stochastic optimization methods, e.g. dynamic *Random Search* [7, 25] and *Differential Evolution* [20]. I has been proposed [23] to use a paired comparison-based *Interactive Differential Evolution* method with faster rates. In *Direct Policy Search*, paired noisy optimization has been proposed in [22, 21, 9]. Our work follows such approaches and combines them with stratified sampling. This is developed in the next section. In *Stochastic Dynamic Programming* (SDP) [1] and its dual counterpart Dual SDP [17], the classical *Sample Average Approximation* (SAA) reduces the problem to a finite set of scenarios; the same set of random seeds is used for all the optimization run. It is indeed often difficult to do better, because there are sometimes not infinitely many scenarios available. Variants of dual SDP have also been tested with increasing set of random realizations [14] or one (new, independent) random realization per iteration [19]. A key point in SDP is that one must take care of anticipativity constraints, which are usually tackled by a special structure of the random process. This is beyond the scope of this paper; we focus on direct policy search, in which this issue is far less relevant as long as we can sample infinitely many scenarios. However, our results on the compared benefits of stratified sampling and common random numbers suggest similar tests in non direct approaches using Bellman values.

2 Algorithms

2.1 Different forms of pairing

For each request x_n to the objective function oracle, the algorithm also provides a set $Seed_n$ of random seeds; $Seed_n = \{seed_{n,1}, \ldots, seed_{n,m_n}\}$. $\mathbb{E}f(x_n, w)$ is then approximated as $\frac{1}{m_n}\sum_{i=1}^{m_n} f(x_n, seed_{n,i})$.

One can see in the literature different kinds of pairing. The simplest one is as follows: all sets of random seeds are equal for all search points evaluated during the run, i.e. $Seed_n$ is the same for all n. The drawback of this approach is that it relies on a sample average approximation: the good news is that the objective function becomes deterministic; but the approximation of the optimum is only good up to the relevance of the chosen sample and we can not guarantee

convergence to the real optimum. Variants consider m_n increasing and nested sets $Seed_n$, such as $\forall(n \in \mathbb{N}^+, i \leq m_n)$, $m_{n+1} \geq m_n$ and $seed_{n,i} = seed_{n+1,i}$. A more sophisticated version is that all random seeds are equal inside an offspring, but they are changed between offspring (see discussion above). We will test this, as an intermediate step between CRN and no pairing at all. In Section 2.2, we explain on an illustrative example why in some cases, pairing can be detrimental. It might therefore make sense to have partial pairing. In order to have the best of both worlds, we propose in Section 2.3 an algorithm for switching smoothly from full pairing to no pairing at all.

2.2 Why common random numbers can be detrimental

The phenomenon by which common random numbers can improve convergence rates is well understood; correlating the noise between several points tends to transform the noise into a constant additive term, which has therefore less impact - a perfectly constant additive term has (for most algorithms) no impact on the run. Setting $\alpha = 1$ in Eq. 5 (below), modelizing an objective function, provides an example in which pairing totally cancels the noise.

$$f(x, w) = ||x||^2 + \alpha w' + 20(1 - \alpha)w'' \cdot x \tag{5}$$

We here explain why CRN can be detrimental on a simple illustrative example. Let us assume (toy example) that

- We evaluate an investment policy on a wind farm.
- A key parameter is the orientation of the wind turbines.
- A crucial part of the noise is the orientation of wind.
- We evaluate 30 different individuals per generation, which are 30 different policies - each individual (policy) has a dominant orientation.
- Each policy is evaluated on 50 different simulated wind events.

With CRN: If the wind orientation (which is randomized) was on average more East than it would be on expectation, then, in case of pairing (i.e. CRN), this "East orientation bias" is the same for all evaluated policies. As a consequence, the selected individuals are more East-oriented. The next iterate is therefore biased toward East-oriented.

Without CRN: Even if the wind orientation is too much East for the simulated wind events for individual 1, such a bias is unlikely to occur for all individuals. Therefore, some individuals will be selected with a East orientation bias, but others with a West orientation bias or other biases. As a conclusion, the next iterate will incur an average of many uncorrelated random biases, which is therefore less biased.

2.3 Proposed intermediate algorithm

We have seen that pairing can be efficient or detrimental depending on the problem. We will here propose an intermediate algorithm (Algorithm 1), somewhere

Algorithm 1 One iteration of a population-based noisy optimization algorithm with pairing.

Require: A population-based noisy optimization algorithm (in particular, rule for generating offspring)
Require: n: current iteration number
Require: $r \in \mathbb{N}^+$: a resampling rule
Require: λ: a population size
Require: $g : \mathbb{N}^+ \to \mathbb{N}^+$: a non-decreasing mapping such that $g(r) \geq r$
1: Generate λ individuals $i_1, \ldots, i_\lambda$ to be evaluated at this iteration
2: Compute the resampling number r by the resampling rule
3: Generate $P_{r,g(r)} = (w_{r,1}, \ldots, w_{r,g(r)})$ a set of $g(r)$ random seeds (we will see below different rules)
4: Each of these λ individuals is evaluated r times with r distinct random seeds randomly drawn in the family $P_{r,g(r)}$.

in between the paired case ($g(r) = r$) and the totally unpaired case ($g(r) >> r$).

The $P_{r,g(r)}$ can be

- Nested, i.e. $\forall (i,r),\ g(r) \geq i \Rightarrow w_{r,i} = w_{r+1,i}$. The $(w_{r,i})_{i \leq g(r)}$ for a fixed r are then independent.
- Independent, i.e. all the $w_{r,i}$ are randomly independently identically drawn.

SAA is equivalent to the nested case with $n \mapsto r(n)$ constant, i.e. we always use the same set of random seeds. [14] corresponds to the nested case. Classical CRN consists in $g(r) = r$ and independent sampling.

We will design, in Section 3, an artificial testbed which smoothly (parametrically depending on α in Eq. 5) switches

- from an ideal case for pairing (testbed in which pairing cancels the noise, as $\alpha = 1$ in Eq. 5);
- to worst case for pairing (counterexample as illustrated above, Section 2.2).

and which (depending on $g(\cdot)$) switches from fully paired to fully independent. We will compare stratified sampling and paired sampling on this artificial testbed. Later, we will consider a realistic application (Section 4).

3 Artificial experiments

We consider a $(\mu/\mu, \lambda)$-Self-Adaptive Evolution Strategy, with $\lambda = 8d^2$, $\mu = min(2d, \lambda/4)$ and some resampling rule $r(n) = \lceil n^d \rceil$, where n is the current iteration number. We apply Algorithm 1 with $g : \mathbb{N}^+ \mapsto \mathbb{N}^+$ defined by

$$g(r) = round(r^\beta),$$

where $\beta \geq 1$ is a parameter which regulates the pairing level. When $\beta = 1$, the function evaluations are fully paired; when $\beta \to \infty$, the function evaluations are fully independent. All experiments are performed with 10000 function evaluations and are reproduced 9999 times.

3.1 Artificial testbed for paired noisy optimization

With $w = (w', w'')$, let us define

$$f(x, w) = ||x||^2 + \alpha w' + 20(1 - \alpha) w'' \cdot x \tag{5}$$

where $\cdot$ denotes the scalar product. Two different cases are considered for the random processes:

- *Continuous case:* w' is a unidimensional standard Gaussian random variable and w'' is a d-dimensional standard Gaussian random variable.
- *Discrete case:* w' is a Bernoulli random variable with parameter $\frac{1}{2}$ and w'' is a vector of d independent random variables equal to 1 with probability $\frac{1}{2}$ and -1 otherwise. For the stratified sampling, in case of 4 strata, we use the 2 first components of w'', which lead to 4 different cases: one for $(-1, -1)$, one for $(1, 1)$, one for $(-1, 1)$ and one for $(1, -1)$.

The motivations for this testbed are as follows:

- It is a generalization of the classical sphere function.
- The case $\alpha = 1$ is very easy for pairing (just a *Sample Average Approximation* (SAA) is enough for fast convergence as in the noise-free case - $\beta = 1$, i.e. $g(r(n)) = r(n)$, leads to canceling noise, even with resampling number $r(n) = 1$).
- The case $\alpha = 0$ is very hard for pairing; the case $\beta = 1$ (full pairing) means that the noise has the same bias for all points.
- For the discrete framework, the stratified sampling directly reduces the dimension of the noisy case: the two first components have no more noise in the stratified case.

3.2 Experimental results

We study

$$\mathbb{E} \frac{\log ||x||^2}{\log n_e} \tag{6}$$

(the lower the better), where x is the estimate of the optimum after $n_e = 10000$ function evaluations and the optimum is 0. Experiments are reproduced 9999 times. The continuous case leads to results in Table 1. Standard deviations are ± 0.0015 for the worst cases and are not presented. Essentially, the results are:

- When α is close to 1, small β (more pairing) is better.
- When α is close to 0, large β (nearly no pairing) is better.

In the discrete case, it is easy to define pairing: we can use strata correspond to distinct values of the two first components of w''. Using the four strata corresponding to the 2 possible values of each of the two first components of w'', we get results presented in Table 2. We still see that pairing is good or bad depending on the case (sometimes leading to no convergence whereas the non-paired case converges, see row $\alpha = 0$ in dimension 5) and never brings huge improvements; whereas stratified sampling is always a good idea in our experiments.

Table 1. Efficiency (average values) of pairing (i.e. case β small) in the continuous case. Left hand side columns (β small) have more pairing than right hand side columns. Pairing is efficient for the "gentle" noise $\alpha = 1$, up to a moderate 50% faster; but it is harmful when $\alpha = 0$ (correlated noise). Next results will investigate stratification. Bold font shows best performance and significant improvements. Positive numbers correspond to no convergence; they are never in bold. Intermediate values of β (intermediate levels of pairing) were never significantly better than others and not clearly more robust to changes in α.

α	$\beta = 1.0$ (paired)	$\beta = 1.16$	$\beta = 1.35$	$\beta = 1.57$	$\beta = 1.82$	$\beta = 2.12$	$\beta = 2.46$ ($\simeq$ unpaired)
dimension 2 (bold for best tested algorithm)							
$\alpha = 0$	-0.07435	-0.06654	-0.07670	-0.08581	-0.09219	**-0.09603**	-0.09344
$\alpha = 0.8$	-0.34475	-0.34661	-0.35921	-0.36253	-0.36565	-0.36709	**-0.36917**
$\alpha = 1$	**-0.75048**	-0.52772	-0.50544	-0.49794	-0.49109	-0.49339	-0.49182
dimension 3 (bold for best tested algorithm)							
$\alpha = 0$	-0.06258	-0.06373	-0.07978	-0.09489	-0.10463	-0.10931	**-0.10977**
$\alpha = 1$	**-0.47681**	-0.43320	-0.41439	-0.41004	-0.40880	-0.40202	-0.39641
dimension 5 (bold for best tested algorithm)							
$\alpha = 0$	0.02965	0.03964	0.04409	0.04394	0.04680	0.04826	0.04823
$\alpha = 0.8$	-0.15077	-0.15977	-0.16369	-0.16687	-0.16770	-0.16793	**-0.16920**
$\alpha = 1$	**-0.23235**	-0.23188	-0.23174	-0.23125	-0.23225	-0.23232	-0.23182
Dimension 10 (bold for best tested algorithm)							
	$\beta = 1$ (paired)						$\beta = \infty$ ($\simeq$ unpaired)
$\alpha = 0$	0.097						**-0.033**
$\alpha = 0.8$	0.038						**-0.053**
$\alpha = 1.0$	**-0.057**						-0.054

4 Real world experiments

4.1 Paired noisy optimization for dynamic problems

Paired statistical tests (e.g. Pegasus [5]) convert a stochastic optimization problem into a deterministic and easier one. Although Pegasus can cause excessive "overfitting" (specialization to the set of considered seeds) when using a fixed number of scenarios, several methods, e.g. using *Wilcoxon signed rank sum test* or changing the scenarios during learning, can reduce the "overfitting" [22, 21]. *Wilcoxon signed rank sum test* pays more attention to small improvements across all scenarios rather than large changes over the return of an individual one, so that it can reduce the "overfitting" caused by a few extreme (good or bad) scenarios. [21] also shows that using an adaptive number of trials for each policy can speed-up learning in such a CRN framework. In the present work, we use new scenarios for each generation - we assume that there is no constraint on the availability of possible realizations w. Another related existing work is [9]. It compares *Independent Random Numbers* (IRN), *Common Random Numbers* (CRN) and *Partial Common Random Numbers* (PCRN, which use pairing in the sense that the same pseudo-random numbers are used several times but in different orders) for *Simultaneous Perturbation Stochastic Approximation* and *Finite Differences Stochastic Approximation*. Both algorithms are faster when using CRN. The present work is dedicated to evolution strategies.

Table 2. Table of results (average slope as in Eq. 6; the lower, the better) depending on α (defining the problem) and β (defining the level of pairing; $\beta = 1$ means full pairing, β large means no pairing). We see that pairing can have a positive or a negative effect. We include results with stratified sampling; which are better or much better depending on the cases. Negligible standard deviations are not presented. Numbers in the stratified case are in bold when they outperform the non stratified setting.

α	$\beta = 1.0$	$\beta = 1.16$	$\beta = 1.35$	$\beta = 1.57$	$\beta = 1.82$	$\beta = 2.12$	$\beta = 2.46$
dimension 2, no stratified sampling (bold for signif. best)							
$\alpha = 0$	-0.07200	-0.06392	-0.07926	-0.08873	-0.09539	-0.09443	-0.09382
$\alpha = 1$	-0.74716	-0.52659	-0.50665	-0.49758	-0.49383	-0.49402	-0.49310
dimension 3, no stratified sampling (bold for signif. best)							
$\alpha = 0$	-0.00802	-0.00519	-0.01246	-0.01672	-0.01750	-0.01660	-0.01635
$\alpha = 0.4$	-0.09327	-0.10422	-0.11704	-0.12771	-0.13248	-0.13375	-0.13138
$\alpha = 0.8$	-0.25365	-0.27016	-0.28168	-0.29045	-0.29341	-0.29459	-0.29474
$\alpha = 1$	-0.39480	-0.38398	-0.37981	-0.37504	-0.37562	-0.37646	-0.37653
dimension 3, stratified sampling (bold if better than no stratification)							
$\alpha = 0$	**-0.01931**	**-0.01396**	**-0.02585**	**-0.03590**	**-0.04430**	**-0.04836**	**-0.04744**
$\alpha = 0.8$	**-0.26548**	**-0.28079**	**-0.29481**	**-0.30133**	**-0.30797**	**-0.30761**	**-0.30763**
$\alpha = 1$	**-0.39714**	-0.38346	**-0.38021**	**-0.37749**	-0.37411	-0.37614	-0.37442
dimension 5, no stratified sampling (bold for signif. best)							
$\alpha = 0$	0.03285	0.04253	0.04896	0.04962	0.05125	0.05336	0.05412
$\alpha = 1$	-0.23188	-0.23207	-0.23265	-0.23080	-0.23219	-0.23148	-0.23042
Dimension 5, stratified sampling (bold if better than no stratification)							
$\alpha = 0$	0.00197	-0.00880	-0.02657	-0.04158	-0.04991	**-0.05404**	-0.04617
$\alpha = 1$	**-0.23294**	-0.23146	-0.23161	**-0.23150**	**-0.23228**	**-0.23158**	**-0.23198**
Dimension 10, no stratified sampling (bold for signif. best)							
	$\beta = 1$ (paired)						$\beta = \infty$ ($\simeq$ unpaired)
$\alpha = 0$	0.108						-0.105
$\alpha = 0.8$	0.012						-0.072
$\alpha = 1$	-0.056						-0.055
Dimension 10, stratified sampling (bold if better than no stratification)							
$\alpha = 0$	0.047						**-0.106**
$\alpha = 0.8$	**-0.033**						-0.072
$\alpha = 1$	**-0.057**						**-0.056**

4.2 Unit commitment problem

For real world experiments, we consider the following sequential decision making problem in the *Markov Decision Processes* (MDP) framework, using discrete time steps: 10 batteries are managed to store energy bought and sold on the electricity market and 10 decision variables have to be made at each time step (i.e. the quantity of energy to buy or to sell for each battery) in order to maximize profits. We apply *rolling planning*, also known as *shrinking horizon*, i.e. new forecasts are used for updating the decisions. There are 168 time steps, i.e. 7 days with one hour per time step. We use an *operational horizon* $o = 5$ time steps, i.e. decisions are made by groups of 5 time steps. When a decision is made, it covers 5 decisions and there is no recourse on these decisions. We have a *tactical horizon* $h = 10$ time steps, i.e. we optimize over the 10 next time steps to speed up computations instead of doing it for all remaining time steps.

4.3 Testbed

We define the following variables: x is the vector of the weights of a neural network; x parametrizes the energy policy described in Eqs. 7 and 8 and d is the

dimension of x. w is a random process modeling the market price. The policy (Eq. 7) uses a neural network to decide the parameters (Eq. 8) of the valorization function. The valorization function provides an estimate of the marginal value of each stock; that is, it provides, for each stock, how much (on the reward over the tactical horizon) we are willing to pay for increasing this stock by one unit.

$$d_t = \arg\max(\ reward\ over\ (t, \ldots, t+h)\ ) + \sum_{i=1}^{d'} \zeta_i s_{t+h,i}. \tag{7}$$

Each state variable corresponds to a stock. We see in Eq. 7 a compromise between the current reward (first term) and the sum $\sum_{i=1}^{10} \zeta_i \times s_{t+h,i}$ over stocks (second term). The ζ_i are estimates of the marginal values of each stock by the neural network. In Eq. 7, d_t is the vector of decisions to apply from the current time step t to time step $t+h$; $s_{t+h} = (s_{t+h,1}, \ldots, s_{t+h,d'})$ is the state at the end of the tactical horizon (the quantity of energy contained in each of the 10 batteries); d' is the number of outputs of the neural network. It is equal to the number of stocks, as we have one marginal value per stock. ζ_i is the i^{th} output of the neural network:

$$(\zeta_1, \ldots, \zeta_{d'}) = neuralNetwork(x, s_t). \tag{8}$$

$s_{t+h,i}$ depends on the random process and the decision:

$$(reward_t, s_{t+h}) = transition(s_t, d_t, random\ process). \tag{9}$$

$reward_t$ is the reward over the operational horizon, i.e. from time t to $t+o$, i.e. $t+5$. The *transition* function describes the problem. We use a (μ, λ)-evolution strategy to optimize x according to the objective function $f(x, w)$. $f(x, w)$ is the simulation function: it applies repeatedly the policy (Eq. 7) and the *transition* function (Eq. 9) from an initial state s_0 to a final state s_{168}. The returned value is the cumulative reward, i.e. the sum of the $reward_t$. The following setup is used: $d = 60$; $\lambda = 4(d+1) = 244$; $\mu = \lambda/4$; $r(n) = \lceil 10\sqrt{n+1} \rceil$. We define paired optimization (a.k.a common random numbers) and stratified sampling in such a case:

- We apply an evolutionary algorithm for optimizing the parameters (i.e. the weights) $x = (x_1, \ldots, x_{60})$ of the neural network controller.
- Each evaluation is a Monte Carlo average reward for a vector of parameters; a Monte Carlo evaluation is a call to $f(x, w)$ above.
- These evaluations are either pure Monte Carlo, paired Monte Carlo, stratified Monte Carlo or paired stratified Monte Carlo.

Common random numbers for energy policies: In the case of CRN (also known as pairing) for the specific case of energy policies, we apply $g(r(n)) = r(n)$, i.e. the same random outcomes $w_1, \ldots, w_{r(n)}$ are used for all individuals of a generation. The random outcomes $w_1, \ldots, w_{r(n)}$ are independently drawn for each new generation.

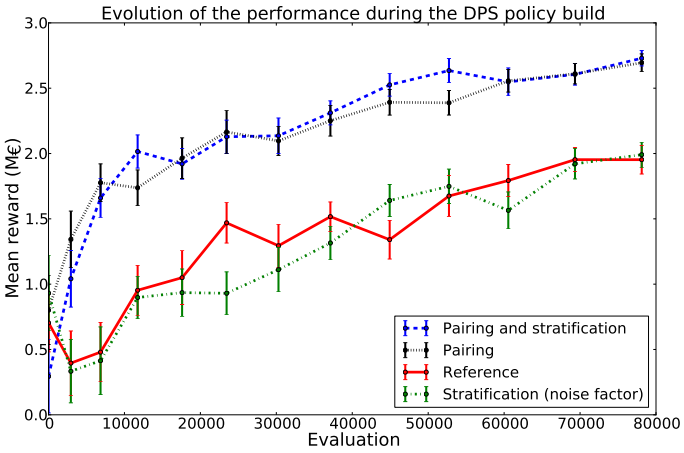

Fig. 1. X-axis: evaluation index. Y-axis: reward (the higher the better). We see that pairing is very efficient whereas stratification provides no clear improvement.

Stratified sampling for energy policies: Stratification in the general case was defined earlier; we here discuss the application to our specific problem. It is very natural, as far as possible, to ensure that points are equally sampled among the 25% best cases, the 25% worst cases, the second quartile and the third quartile.

Even if these categories can only be approximately evaluated, this should decrease the variance. It is usually a good idea to stratify according to quantiles of a quantity which is as related as possible to the quantity to be averaged, i.e. $f(x, w)$. The four strata are the four quantiles on the annual average of an important scalar component of the noise.

Experimental results in Figure 1 show that pairing provides huge improvement in the realistic case. Stratification has a minor impact.

5 Conclusions

We tested, in an artificial test case and a Direct Policy Search problem in power management, paired optimization (a.k.a common random numbers) and partial variants of it. We also tested stratified sampling. Both algorithms are easy to implement, "almost" black-box and applicable for most applications. Paired optimization is unstable; it can be efficient in simple cases, but detrimental with more difficult models of noise, as shown by results on $\alpha = 1$ (positive effect) and $\alpha = 0$ (negative effect) in the artificial case (Eq. 5). We provided illustrative examples of such a detrimental effect (Section 2.2). Stratification had sometimes a positive effect on the artificial test case and was never detrimental. Nonetheless, on the realistic problem, pairing provided a great improvement, much more than stratification. Pairing and stratification are not totally black

box; however, implementing stratification and pairing is usually easy and fast and we could do it easily on our realistic problem. We tested an intermediate algorithm with a parameter for switching smoothly from fully paired noisy optimization to totally unpaired noisy optimization. However, this parametrized algorithm (intermediate values of β) was not clearly better than the fully unpaired algorithm ($\beta = \infty$). It was not more robust in the case $\alpha = 0$, unless β is so large that there is essentially no pairing at all. As a conclusion, we firmly recommend common random numbers for population-based noisy optimization. Realistic counter-examples to CRN's efficiency would be welcome - we had such detrimental effects only in artificially built counter-example. There are probably cases (e.g. problems with rare critical cases) in which stratification also helps a lot, though this was not established in our application (which does not have natural strata).

Further work. Other variance reduction techniques are possible. A nice challenge for future research is to find algorithms protecting variance reduction techniques from their possible detrimental effects (e.g. as efficient as CRN when $\alpha = 1$ in Eq. 5 and as efficient as no pairing when $\alpha = 0$). In particular importance sampling with optimal allocation per stratum (though we need variance estimates for that, which is difficult in a noisy optimization setting), quasi Monte Carlo (more difficult in a nearly black-box setting), or quantization [4, 6].

Also, we used $g(r) = round(r^\beta)$. Results were somehow disappointing. Maybe more subtle formulas, with $g(r) = round(Ar^B)$, could be used instead, in particular $B = 1$ and $A > 1$; or $g(r)$ might be made adaptive.

References

1. Bellman, R.: Dynamic Programming. Princeton Univ. Press (1957)
2. Billingsley, P.: Probability and Measure. John Wiley and Sons (1986)
3. Cranley, R., Patterson, T.: Randomization of number theoretic methods for multiple integration. SIAM J. Numer. Anal. 13(6), 904914 (1976)
4. Defourny, B.: Machine Learning Solution Methods for Multistage Stochastic Programming. Ph.D. thesis, Institut Montefiore, Université de Liège (2010)
5. Dowell, M., Jarratt, P.: The "pegasus" method for computing the root of an equation. BIT Numerical Mathematics 12(4), 503–508 (1972), http://dx.doi.org/10.1007/BF01932959
6. Dupacov, J., Gröwe-Kuska, N., Römisch, W.: Scenario reduction in stochastic programming: An approach using probability metrics. No. 20 in Stochastic Programming E-Print Series, Institut fr Mathematik (2000), \url{http://edoc.hu-berlin.de/docviews/abstract.php?id=26613}, published; Springer; Berlin [u.a.]; Mathematical Programming; 95; 2003; 3
7. Hamzaçebi, C., Kutay, F.: Continuous functions minimization by dynamic random search technique. Applied Mathematical Modelling 31(10), 2189–2198 (2007)
8. Huang, S.C., Coulom, R., Lin, S.S.: Monte-Carlo simulation balancing in practice. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games. Lecture Notes in Computer Science, vol. 6515, pp. 81–92. Springer (2010)
9. Kleinman, N.L., Spall, J.C., Naiman, D.Q.: Simulation-based optimization with stochastic approximation using common random numbers. Management Science

45(11), 1570–1578 (1999), http://pubsonline.informs.org/doi/abs/10.1287/mnsc.45.11.1570
10. Kozak, M.: Optimal stratification using random search method in agricultural surveys. Statistics in Transition 6(5), 797–806 (2004), http://www.researchgate.net/publication/229051808_Optimal_stratification_using_random_search_method_in_agricultural_surveys/file/d912f5062bc010dd58.pdf
11. Lavallée, P., Hidiroglou, M.: On the stratification of skewed populations. Survey Methodology 14(1), 33–43 (1988), http://www.amstat.org/sections/srms/Proceedings/papers/1987_142.pdf
12. Linderoth, J., Shapiro, A., Wright, S.: The empirical behavior of sampling methods for stochastic programming. Annals OR 142(1), 215–241 (2006), http://dblp.uni-trier.de/db/journals/anor/anor142.html#LinderothSW06
13. Mascagni, M., Chi, H.: On the scrambled halton sequence. Monte-Carlo Methods Appl. 10(3), 435–442 (2004)
14. de Matos, V., Philpott, A., Finardi, E.: Improving the performance of stochastic dual dynamic programming. Applications – OR and Management Sciences (Scheduling) (2012), http://www.optimization-online.org/DB_FILE/2012/07/3529.pdf
15. Morokoff, W.J.: Generating quasi-random paths for stochastic processes 40(4), 765–788 (Dec 1998), http://epubs.siam.org/sam-bin/dbq/article/31795
16. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods (1992)
17. Pereira, M.V.F., Pinto, L.M.V.G.: Multi-stage stochastic optimization applied to energy planning. Math. Program. 52(2), 359–375 (Oct 1991), http://dx.doi.org/10.1007/BF01582895
18. Sethi, V.: A note on optimum stratification of populations for estimating the population means. Australian Journal of Statistics 5(1), 20–33 (1963)
19. Shapiro, A., Tekaya, W., da Costa, J.P., Soares, M.P.: Risk neutral and risk averse stochastic dual dynamic programming method. European Journal of Operational Research 224(2), 375–391 (2013)
20. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4), 341–359 (1997)
21. Strens, M., Lx, H.G., Moore, A., Brodley, E., Danyluk, A.: Policy search using paired comparisons. Journal of Machine Learning Research 3, 921–950 (2002)
22. Strens, M., Moore, A.: Direct policy search using paired statistical tests. In: Proceedings of the 18th International Conference on Machine Learning. pp. 545–552. Morgan Kaufmann, San Francisco, CA (2001)
23. Takagi, H., Pallez, D.: Paired comparison-based interactive differential evolution. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. pp. 475–480. IEEE (2009)
24. Wang, X., Hickernell, F.: Randomized halton sequences. Math. Comput. Modelling 32, 887–899 (2000)
25. Zabinsky, Z.B.: Random search algorithms. Wiley Encyclopedia of Operations Research and Management Science (2009)