DCMA, yet another derandomization in Covariance-Matrix-Adaptation

O. Teytaud Tao, Inria, Lri, Umr-Cnrs-8623
bat 490 Univ. Paris-Sud 91405 Orsay Cedex
France
teytaud@lri.fr

S. Gelly Tao, Inria, Lri, Umr-Cnrs-8623
bat 490 Univ. Paris-Sud 91405 Orsay Cedex
France
gelly@lri.fr

ABSTRACT

In a preliminary part of this paper, we analyze the necessity of randomness in evolution strategies. We conclude to the necessity of "continuous"-randomness, but with a much more limited use of randomness than what is commonly used in evolution strategies. We then apply these results to CMA-ES, a famous evolution strategy already based on the idea of derandomization, which uses random independent Gaussian mutations. We here replace these random independent Gaussian mutations by a quasi-random sample. The modification is very easy to do, the modified algorithm is computationally more efficient and its convergence is faster in terms of the number of iterates for a given precision.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization, Unconstrained optimization*
; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: [Numerical Algorithms and Problems]

General Terms

Algorithms, Theory

Keywords

Evolution Strategies, Derandomization

1. INTRODUCTION

Evolution Strategies (ES)[17, 22, 7] are an important stream of evolutionary algorithms with usually the following elements:

- they are derivative-free and in most cases only depend on comparisons between fitness values (as well as direct search methods, see e.g. [6]);
- they usually work in numerical optimization in continuous domains;
- their crossovers are usually restricted to weighted averages of selected individuals;
- they are usually more robust than tools from mathematical programming - but it must be emphasized that there are not a lot of large-scale comparisons with direct search methods yet, in particular on benchmarks which are not the usual ones for evolution strategies (see e.g. [9] for a set of problems that is usual in direct search methods);
- they have more theoretical foundations than many fields of evolutionary algorithms (see e.g. [3] for a survey of theoretical analysis of evolution strategies).

CMA-ES (Covariance-Matrix-Adaptation-ES) is a well known evolution strategy. The reader is referred to [11] for details, but very roughly, its principle is as follows:
1. randomly generate a population of p points according to a Gaussian distribution;
2. depending on the fitness of these points, update the parameters of the Gaussian distribution so that the distribution gets closer to the distribution of best points in the population;
3. go back to step 1.

In this paper, we improve two important qualities of, respectively, CMA-ES and ES in general, by the use of quasi-random points.

First, the main advantage of CMA-ES comparing to other evolution strategies is the derandomization of the search direction, estimating an ellipsoid containing the best points. However, it is known that the conditioning of the points used for building an approximation of the fitness is important. This conditioning can be improved by choosing well-distributed points instead of random points. The covariance-matrix-adaptation being averaged among multiple epochs, the distribution of points must be nice not only within a given offspring but also among multiple generations. This is handled by quasi-random points, as the offspring at one epoch will be "not-too-redundant" with the previous off-springs. This is the first advantage of our derandomization below.

Second, Evolution Strategies, thanks to random exploration, more carefully explore the fitness-landscape than many other optimization methods. We improve this exploration by the use of quasi-random points, in the same manner as quasi-random-search ([15]) outperforms random-search. This is the second advantage of our derandomization below.

GECCO '07 London

Quasi-random experiments in evolutionary computation have already been performed in [13]; we here use:

- a more sophisticated evolutionary computation algorithm,
- a low-discrepancy sequence more suitable for high-dimension thanks to scrambling,
- a relation between quasi-random successive offsprings that is particularly easy thanks to the fact that only one Gaussian is used to generate an offspring. This is the case typically for $(\mu/\mu, \lambda)$-evolution strategies (with or without weighting of averages), and CMA-ES in particular. We here focus on this simpler case as this family of algorithms is already very efficient (e.g. on the Cec'05 benchmark[21]), and widely used in practice.

So, inspired by [13, 2, 1], we here replace the random generation of step 1 by a quasi-random generation ([15, 16, 23, 14]). Note in particular that modern quasi-random methods are compatible with high-dimensionality ([19]), especially thanks to *scrambling* (mixing of deterministic-quasi-random with randomness[14]). We here use scrambling-Halton sequences ([10, 18]) and Sobol sequences. The resulting algorithm is termed "DCMA", for Derandomized-CMA. The generation of quasi-random Gaussian points is straightforward from classical random Gaussian points and will be detailed in section 3.

Our main new derandomized algorithm is termed DCMA (Derandomized-CMA) and uses quasi-random Gaussian mutations instead of random Gaussian mutations as in CMA. This is a very light modification of the source code, and it is a very stable improvement as shown in experimental sections above. We also consider DdCMA, Derandomized-distance-CMA, which consists in quasi-randomizing only the step size (the direction remaining random), and not the whole mutation-vector like in DCMA. In DdCMA, the mutation (before covariance and step-size modification) is $qrand \times rand$, where $qrand$ is a quasi-random uni-dimensional Gaussian number (see section 3) and $rand$ is a usual unit random vector uniform in the sphere.

The paper is organized as follows. Section 2 presents some theoretical insights about the use of randomness and in particular the limits of derandomization in terms of robustness. Section 3 presents Gaussian quasi-random points. Section 4 presents the experimental comparisons.

2. A BIT OF THEORY: WHY RANDOMNESS IS USEFUL ?

Some papers have been devoted to the derandomization of evolution strategies ([13, 1, 11, 8]). We here study theoretically the limits of the derandomization of evolution strategies, and propose in next section a derandomization that is in particular relevant for the Covariance-Matrix-Adaptation algorithm (and also in a straightforward manner for various $(1+\lambda)$, $(1, \lambda)$ or $(\mu/\mu, \lambda)$ evolution strategies).

In this section, we show that (i) randomization is necessary to ensure almost sure convergence to the essential minimum of a fitness function on a continuous domain (ii) only one random uniform number in $X = [0, 1]^d$ is enough (without any non-smooth creation of many real numbers from this vector). The positive result with one random uniform number is reached using a standard quasi-random sequence with a simple classical randomization by random shift.

An optimization algorithm is defined in the formalism of machines-with-oracle (the oracle is the fitness-function) as described in figure 1. X is the domain, X'_n is the estimate of the optimum by the algorithm, X_n is the point that the algorithm decides to visit (i.e., the point of X for which it requests an answer from the Oracle, i.e. for which it requests a fitness-value). In many cases, but not necessarily, $X'_n = X_{i_n}$ with $i_n = \arg\min_{i \in [[1,n]]} Oracle(X_i)$.

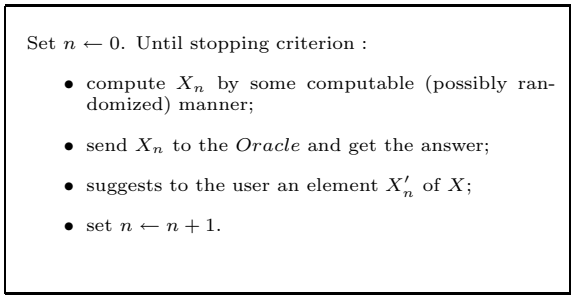

Figure 1: Formalization of optimization algorithms on a domain X.

One says that the algorithm converges if $Oracle(X'_n)$ converges to the minimum of $Oracle$, where X'_n is the sequence of elements suggested by the algorithm.

THEOREM 2.1 (FINDING MINIMA IS NOT POSSIBLE). *We consider X infinite and uncountable (typically, $[0, 1]$). Consider A an optimization algorithm (possibly stochastic), providing elements of X and requesting answers from a function Oracle. Then, there exists a uncountable set of deterministic functions such that for each of these functions, A almost surely does not converge.*

Proof:

Consider $X_n^{x \mapsto 1}$ the (possibly) stochastic sequence of requests made by A if all answers of the function $Oracle$ are 1. Define $P_{n,u}$ for $u \in X$ the probability of $X_n^{x \mapsto 1} = u$. Then,

$$\sum_u P_{n,u} \leq 1.$$

Any family of positive real numbers with finite sum has at most a countable set of non-zeros values. So, for each n, the set P_n of u such that $P_{n,u} > 0$, namely

$$P_n = \{u; P_{n,u} > 0\}$$

is countable.

The set of u such that $\exists n; P_{n,u} > 0$ is a countable union (indexed by n) of countable sets (the P_n's). Hence, this set is countable as a countable union of countable sets. Consider V the complementary set:

$$V = \{u \in X \text{ such that } \forall n, P_{n,u} = 0\}.$$

V is uncountable.

Then, by definition, any element of V has a probability 0 of being provided to the $Oracle$, as long as the $Oracle$ replies 1.

Let $(pb_v)_{v \in V}$ the family of functions defined by

$$pb_v(u) = 0 \text{ if } u = v$$

and

$$pb_v(u) = 1 \text{ if } u \neq v$$

This family is then infinite and uncountable. For $v \in V$, we now consider what happens if we apply A to pb_v.

For any n fixed, if $\forall i < n, X_i \neq v$, then $P(X_n = v) = 0$ by definition of V. Hence,

$$P(X_n = v) \leq \sum_{i=1}^{n-1} P(X_i = v)$$

Then, by induction on n,

$$\forall n, P(X_n = v) = 0.$$

Hence $P(\exists n, X_n = v) = 0$ because countable union of sets of probability 0 is of probability 0. Then the algorithm A has the same behavior on all the problems pb_v with probability 1.

For any $n \geq 1$, consider now the probability $Q_{n,w}$ that the algorithm suggests $X'_n = w$ at step n. For all $n \geq 1$, $\sum_w Q_{n,w} \leq 1 < \infty$. Then $Q_{n,w} > 0$ for at most a countable set V_n of values w. Hence $\cup_{n\in\mathbb{N}} V_n$ is countable. Consider

$$V' = V \setminus (\cup_{n\in\mathbb{N}} V_n)$$

V' is uncountable. Therefore, the set of pb_v for $v \in V'$ is not countable. If we apply A to pb_v for $v \in V'$ then $\forall n, P(X'_n = v) = 0$, hence

$$P(\exists n X'_n = v) = 0.$$

Hence, with probability 1, A does not converge to the optimum. □

We have shown that finding minima is not possible in all cases, even asymptotically. However, we now show that we can find essential minima (i.e. the minimum after removal of negligible sets for Lebesgue's measure).

THEOREM 2.2 (RANDOM FINDS ESSENTIAL MINIMA). *Consider $X = [0,1]^d$. Then, random search as follows:*

- *X_n is independently, uniformly distributed on X;*
- *$X'_n = X_{i_n}$ where $i_n = \arg\min_{i\in[[1,n]]} Oracle(X_i)$*

almost surely converges to the essential minimum of Oracle on X, i.e.

$$Oracle(X'_n) \to \inf\{x; P(Oracle(X_1) \leq x) > 0\}$$

Moreover:

$$P(Oracle(X'_n) > x) \leq P(Oracle(X_1) > x)^n.$$

The straightforward proof is omitted.

But can we ensure the same result without randomization, or with a countable seed ? As shown in the following theorem, the answer is essentially no.

THEOREM 2.3 (LOWER-BOUNDING RANDOMNESS). *Consider $X = [0,1]^d$ and any optimization algorithm (as formalized above), deterministic as a function of the Oracle and of a random seed $w \in \mathbb{N}$ (for any distribution of probability of $w \in \mathbb{N}$). Then, there exists an uncountable family of problems such that $Oracle(X'_n)$ does not converge to the essential minimum of Oracle.*

Proof:

Consider $X_n^{x\mapsto 1,w}$ the sequence of requests made by A if all answers of the *Oracle* are 1 and if the random seed is w.

Consider

$$E = \{X_n^{x\mapsto 1,w}; w \in \mathbb{N}, n \in \mathbb{N}\} \cup \{X'^{x\mapsto 1,w}_n; w \in \mathbb{N}, n \in \mathbb{N}\} \quad (1)$$

the set of all possibly visited or suggested points when the fitness function is the function that always replies 1.

Consider $Oracle(x) = 1$ if $x \in E$ and $Oracle(x) = 0$ otherwise. Then, for any n, $Oracle(X_n) = 1$ and $Oracle(X'_n) = 1$.

As E is countable, the essential minimum of *Oracle* is 0. Therefore, the optimization algorithm does not converge to the essential minimum. □

Therefore, we have shown

- that finding the minimum on a continuous domain is too hard as a goal (theorem 2.1);
- that finding the essential minimum is possible by simple random search (theorem 2.2);
- that finding the essential minimum is not possible without at least an uncountable randomization (theorem 2.3). One could indeed refine theorem 2.3 by extending it to algorithms that have an uncountable random seed but only use a finite number of bits before any iterate (what is not the case for random search as in the proof of theorem 2.2); the proof is essentially the same, i.e. uses countability of E as defined in equation 1.

We are now going to study more carefully the limit of the quantity of randomization required for the convergence to the essential minimum on a continuous domain. Of course, one random number in $[0,1]$ can be expanded in a sequence of numbers in $[0,1]^d$; therefore, theoretically, one random uniform number in $[0,1]$ is enough for generating random search. We will here show that one random seed in $[0,1]^d$ is enough in a smooth manner, namely all the X_n being linear (with congruence modulo 1) as a function of the random seed in $[0,1]^d$:

THEOREM 2.4 (ONE RANDOM SHIFT IS ENOUGH). *Consider the following quasi-random-search algorithm:*

- *uniformly randomly generate $r \in X = [0,1]^d$;*
- *consider $x_1, x_2, \ldots$ a (deterministic) sequence[1] that is dense in $[0,1]^d$;*
- *for $n \geq 1$*
  - *define $X_n = r + x_n$ (modulo 1);*
  - *define $X'_n = X_{i_n}$ where $i_n = \arg\min_{i\in[[1,n]]} Oracle(X_i)$.*

Then this algorithm converges almost surely to the essential minimum of Oracle.

Proof: Consider any e greater than the essential minimum s of *Oracle*. Consider $Q_e = \{x \in X; Oracle(x) \leq e\}$. By definition of the essential minimum, Q_e has positive measure.

[1] In particular, quasi-random numbers can be used.

Consider r a random variable uniformly distributed in X and S the set of the x_i.

For any $e_n = s+1/2^n$ for $n \in N$, we will show below that:

$$\text{almost surely, } r+S \text{ intersects } Q_{e_n} \tag{2}$$

Equation 2 for all n implies that almost surely also, $r+S$ intersects all the Q_{e_n}, and therefore, almost surely, the algorithm converges to the essential minimum of the fitness. Therefore, we just have to show equation 2 for some $Q = Q_{e_n} = Q_{s+1/2^n}$.

In the sequel, we note $A+B = \{a+b \text{ modulo } 1; a \in A, b \in B\}$, and $A-B = \{a-b \text{ modulo } 1; a \in A, b \in B\}$.

Q has non-zero measure. Therefore, Q has density 1 almost everywhere in Q, by Lebesgue's density theorem. We recall below Lebesgue's density theorem:

THEOREM 2.5. *With A measurable, $A \subset \mathbb{R}^d$, $\epsilon > 0$ and $x \in \mathbb{R}^d$, note*

$$d_\epsilon(x, A) = \frac{\mu(A \cap B(x,\epsilon))}{\mu(B(x,\epsilon))}$$

The density of a measurable set $A \subset \mathbb{R}^d$ at a point x is

$$d(x, A) = \lim_{\epsilon \to 0} d_\epsilon(x, A)$$

Consider a measurable subset A of $\mathbb{R}^d$. Then, almost any $x \in A$ verifies $d(x, A) = 1$.

Consider now a particular x at which Q has density 1, i.e.

$$\mu(B(x,\epsilon) \cap Q) \geq (1-\epsilon'(\epsilon))\mu(B(x,\epsilon)) \tag{3}$$

where $\epsilon'(\epsilon)$ decreases to 0 as $\epsilon \to 0$. Note $\mu(B(x,\epsilon) \cap R)/\mu(B(x,\epsilon))$ the ϵ-density of set R at x. As S is dense, $Q-S$ has ϵ-density lower bounded by $\epsilon'(\epsilon)$. This implies that $Q-S$ has density 1 everywhere. Therefore, $Q-S$ has measure 1.

Therefore, with probability 1, r lies in $Q-S$, and therefore $r+S$ intersects Q.

Hence the expected result. □

This random-shift method is termed Cranley-Patterson rotation [**?**]. Note that the theoretical analysis of the algorithm DCMA defined below can be completed by the study of almost-plateau-functions with a small target area (something close to the Needle-In-A-Haystack function, but in continuous domain), with the simplifying assumption that the parameters of the Gaussian distribution are fixed. This assumption is not true in the case of CMA, as the parameters will change according to random selection in a plateau[2] but this is a first step not-so-far from reality. Essentially, under this assumption, the algorithm reduces to a quasi-random search, and such an analysis has already been published in the case of quasi-random-search (see [15]).

The previous theorems state that derandomization is possible, while one must avoid "too much" derandomization. In spite of the "worst-case analysis" nature of the theory above, it also makes sense in practice as shown in the sequel. The previous theorems about optimization are analogous to mathematical analysis of randomized-quasi-random-sequences for integration ([14]): they emphasize the necessity of some randomness.

[2]Depending on the behavior of the algorithm when fitness-values in the population are equal.

3. GAUSSIAN QUASI-RANDOM POINTS

Quasi-random sequences are a wide area of research with a wide and increasing area of application (3.1). We use the method of Halton (3.3). It is based on the Van Der Corput sequence (3.2). The transformation into Gaussian vectors is presented in (3.4). We apply scrambling, which strongly enhances results.

The global computation time is negligible, and not larger than for classical pseudo-random numbers.

3.1 Quasi-random points

Random points can be very disappointing. They can be distributed in a very non-uniform manner. For example, figure 2 (upper-left) is a random independent sample (uniformly drawn in $[0,1]^2$). Unless you are very lucky, you can not get by chance something as regular as other plots in figure 2. Therefore, in many areas of computer science, better-than-random points have been studied (integration [15], optimization [1], path planning [23], learning [5]).

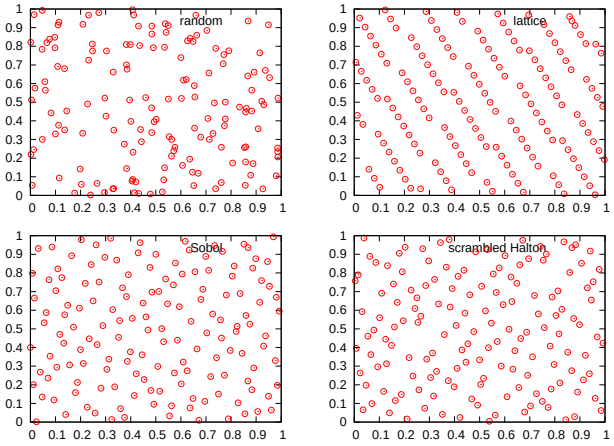

Figure 2: Random points, lattice points, Sobol points, scrambled-Halton points.

We can then define "good" point sets and "bad" point sets. In order to generate good point sets, a measure of goodness would be usefull. Consider a point set $x_1, \ldots, x_n$ in $D = [0,1]^d$. An intuitive idea is $\sup_{x \in D} \inf_{i \in [[1,n]]} d(x, x_i)$ (to be minimized), where d is some distance (e.g. L^∞ distance). Generating points on a grid is easy, and it has been pointed out that this is optimal for the criterion above and for many values of the set size ([23]). However, this point set is not satisfactory. For example, it would be nice that the projection on any axis of a good point set is also a good point set. This is not the case for the lattice in figure 2 or for grids: the projection on some well-chosen axis leads to accumulations (e.g. projection on canonical axis for the standard grid). Therefore, other criterions have been defined. The most well known criterion is discrepancy. There are various discrepancies. The most well known is the following : $\sup_{r \in D} \left| \frac{1}{n} Card\{i \in [[1,n]]; \forall j (x_i)_j \leq r_j\} - \pi_{j \in [[1,d]]} r_j \right|$ with $Card$ the cardinal operator. This formula has an immediate interpretation : it is the largest absolute difference between the area of a rectangle including 0 and the proportion of points in this rectangle. It is much more stable with respect

to projection on an axis. However, it has various drawbacks (see [12]) :

- it only deals with rectangles with axis parallel to the canonical axis;
- it only deals with rectangles;
- it is not symmetric in the sense that the discrepancy of $x_1, \ldots, x_n$ is not the discrepancy of $1 - x_1, \ldots, 1 - x_n$;
- it is a worst case on r.

The two first elements can be debated. If variables are considered in a non-rotation invariant manner, it can be meaningful. The relevance of axis-decomposition is in particular an element in favor of some evolutionary algorithms, which naturally handle partial or total decompositions of objective functions on variables.

The fourth element is probably the main trouble. Fortunately, extensions have already been defined. The main tool is the L^2-star-discrepancy :

$$\sqrt{\int_{r \in D} \left(\frac{1}{n} Card\{i \in [[1,n]]; \forall j (x_i)_j \leq r_j\} - \pi_{j \in [[1,d]]} r_j \right)^2}$$

This form of discrepancy (as well as others) verifies inequalities similar to Koksma's inequality (see [12] on this topic). Many algebraic methods have been defined for generating sequences of points with low discrepancy ([15, 23, 16]). We here focus to Halton's method with a simple scrambling. We present below (i) Van Der Corput quasi-random sequences (that are necessary for defining Halton-sequences) in $[0,1]$, (ii) Halton-sequences in $[0,1]^d$ and a simple scrambling-scheme, (iii) the conversion to Gaussian Vectors.

3.2 Van Der Corput sequence

Consider p a prime number. The following sequence generates the n^{th} element $x_{n,p} \in [0,1]$ of the Van Der Corput sequence in basis p:

- write n in basis p: $n = d_k d_{k-1} \ldots d_1$, i.e. $n = \sum_{i=1}^{k} d_i p^i$ with $d_i \in [[0, p-1]]$;
- $x_{n,p} = 0.d_1 d_2 \ldots d_k$ in basis p, i.e. $x_{n,p} = \sum_{i=1}^{k} d_i p^{-i}$.

A classical improvement, termed *scrambling*, defines $x_{n,p}$ as

$$x_{n,p} = 0.\pi(d_1)\pi(d_2)\ldots\pi(d_k)$$

where π is some permutation of $[[0, p-1]]$ such that $\pi(0) = 0$ in order to ensure $\forall n, x_{n,p} \neq 0$.

3.3 Halton sequence

The Halton sequence generalizes the Van Der Corput sequence to dimension d. Consider p_i the i^{th} prime number. Then, x_n, the n^{th} element of a Halton sequence in dimension d, is

$$x_n = (x_{n,p_1}, x_{n,p_2}, \ldots, x_{n,p_d}) \in [0,1]^d$$

The scrambled-Halton sequence is the use of a randomly drawn permutation for each $i \in [[1,d]]$ (see section 3.2 above).

3.4 Conversion to Gaussian vectors

Quasi-random sequences usually deal with the uniform distribution in $[0,1]^d$: the goal is the approximation of the uniform distribution by finite sets of points. This is also the case of the Halton sequence. To convert a uniform random variable in $[0,1]$ into another random variable X with values in $\mathbb{R}$, the standard solution is to replace $x \in [0,1]$ by x' such that $P(X \geq x') = x$. More generally, the principle of the generation of p quasi-random points according to the Gaussian distribution is as follows for a problem in dimension d (n is initialized at 0 at the beginning of the program) :

Method for generating p quasi-random-Gaussian-points $y_n, \ldots, y_{n+p-1}$ with A and b as parameters of the Gaussian:

- consider p points x_n, x_{n+1}, x_{n+2}, x_{n+p-1} in $[0,1]^d$ by the Halton sequence [10] and update the static variable n by $n \leftarrow n + p$ (this leads to classical quasi-random points in the unit hypercube);
- replace each coordinate of each of these points by its antecedent for the cumulative density function of the Gaussian distribution (this leads to quasi-random points for the standard unit Gaussian distribution);
- possibly: randomly rotate this set of points (with a random rotation in $\mathbb{R}^d$ of centre 0, independently and uniformly randomly drawn at each epoch); doing this adds randomization; when this option is used, the algorithm is termed RDCMA in the sequel;
- if the Gaussian is $AN+b$ where N is the standard unit Gaussian, then multiply each point in the population by A and add b (this leads to quasi-random points for the required Gaussian distribution): $y_{n+k} = Ax_{n+k} + b$.

The result is presented in figure 3 in dimension 2 (with respectively 9, 21, 57 and 153 points).

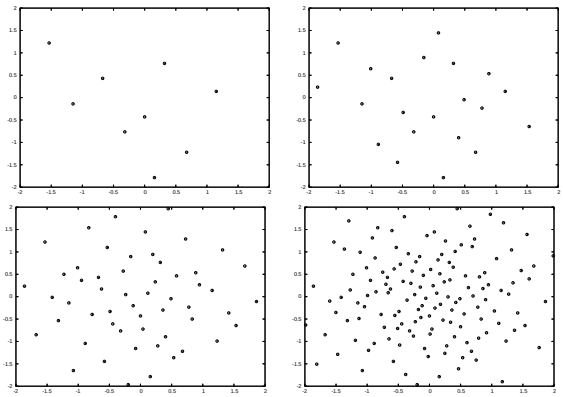

Figure 3: 9, 21, 57 and 153 quasi-random Gaussian points in dimension 2. See text for details.

4. EXPERIMENTAL COMPARISONS

We here compare the performance of the algorithm CMA and several derandomized versions of CMA. As stated in the introduction, our main new derandomized

algorithm is termed DCMA (Derandomized-CMA) and uses quasi-random Gaussian mutations instead of random Gaussian mutations as in CMA. We also consider DdCMA, Derandomized-distance-CMA, which consists in quasi-randomizing only the step size (the direction remaining random). RDCMA uses a random rotation during the offsprings (the rotation is i.i.d. randomly drawn at each epoch).

We consider f the best fitness-value found by the optimization algorithm after a given number of function-evaluations. We report below the mean of $\log(f)$ (natural logarithm) and its standard deviation, the median being within parenthesis, for various classical test functions. All results come from 11 runs of experiments. The initial standard deviation of CMA / DCMA / DdCMA / RDCMA is 1, the covariance matrix is initialized at identity, and the initial point is randomly drawn according to a standard Gaussian distribution. The same starting points were used for CMA and DCMA (and its variants).

Tables in dimension 2, 4, and 6 (figure 4), 8, 10 and 20 (figure 5), 30, 40 and 50 (figure 6), compare CMA and DCMA in various dimensions (average results among 11 runs, with standard deviation and median between parenthesis). All the results concern the logarithm of the fitness. We have also tested the case in which we replace the scrambled-Halton-sequence by a Sobol sequence ([20, 4]), which is known as a good scrambled quasi-random-sequence in large dimensions. The results, included in the extended www-version `http://www.lri.fr/~teytaud/inriadcma.pdf` are not significantly different from those using scrambled-Halton.

Results of RDCMA are not presented; its results are less impressive than those of DCMA, but it outperforms CMA in almost all cases. DdCMA is also not presented; its results are surprisingly bad in particular when the dimension increases. The number of function evaluations allowed, for each algorithm, is $\min(dim \times dim \times 100, 5000)$. The objective functions are those from the Octave/Matlab implementation of CMA-ES (`http://www.bionik.tu-berlin.de/user/niko/cmaes\_inmatlab.html`), but with all constraints removed except box-constraints.

Problem (dim2)	CMA	DCMA
fsphere	-28.2±2.52 (-28.5)	-41±1.98 (-41.)
fsphereoneax	-30.0±5.57 (-30.5)	-41.7±2.87 (-42.0)
frandsphere	-28.2±2.84 (-27.2)	-38.7±0.94 (-38.8)
fspherelb0	-31.2±2.21 (-31.)	-42.±1.36 (-42.3)
fspherehull	-15.0±1.83 (-14.3)	-21.2±1.00 (-21.4)
fsectorsphere	-15.0±2.24 (-15.9)	-21.0±1.80 (-21.5)
fstepsphere	-11.5±13.2 (1e-11)	-20.7±10.2 (-25.3)
flnorm	-13.5±1.82 (-14.2)	-20±0.92 (-21.2)
fbaluja	7.92±1.17 (8.39)	1.19±1.37 (1.21)
fschwefelrosen1	-26.9±2.95 (-26.4)	-40.2±1.87 (-40.)
fschwefelrosen2	-27.0±4.23 (-26.3)	-38.5±0.77 (-38.8)
fconcentric	-3.51±1.11 (-4.14)	-5.70±1.16 (-5.22)
fgriewank	-32.2±2.75 (-32.9)	-48.8±1.83 (-48.)
frastrigin	0.32±0.37 (-0.00)	-11.6±16.9 (-0.00)
Problem (dim4)	CMA	DCMA
fsphere	-35.7±1.50 (-35.3)	-38.5±1.35 (-38.5)
fsphereoneax	-35.4±2.46 (-34.8)	-39.8±2.61 (-40.0)
frandsphere	-33.9±2.04 (-33.2)	-36.0±1.02 (-35.8)
fspherelb0	-36.4±1.24 (-35.8)	-38.2±1.12 (-37.)
fspherehull	-27.1±0.96 (-27.4)	-28.1±0.42 (-28.0)
fsectorsphere	-25.9±3.34 (-26.2)	-34.5±1.68 (-34.1)
fstepsphere	-23.0±7.63 (-25.3)	-25.3±3.72e-15 (-25.3)
flnorm	-26.1±1.09 (-26.5)	-27.6±0.45 (-27.5)
fbaluja	-1.39±1.14 (-1.42)	-5.21±0.47 (-5.11)
fschwefelrosen1	-35.3±0.75 (-35.4)	-38.6±0.60 (-38.6)
fschwefelrosen2	-36.0±1.89 (-36.2)	-37.±1.15 (-37.9)
fconcentric	-2.41±1.23 (-2.90)	-3.24±1.07 (-3.46)
fgriewank	-35.6±2.92 (-35.1)	-42.0±12.4 (-45.4)
frastrigin	1.30±0.86 (1.38)	0.84±0.68 (0.68)
Problem (dim6)	CMA	DCMA
fsphere	-34.±0.85 (-34.9)	-37.±1.06 (-37.8)
fsphereoneax	-35.7±2.16 (-35.0)	-37.2±2.8 (-37.2)
frandsphere	-29.6±6.14 (-31.2)	-34.5±0.93 (-34.7)
fspherelb0	-34.8±0.62 (-34.8)	-37.8±0.67 (-37.6)
fspherehull	-27.5±0.33 (-27.4)	-27.9±0.22 (-27.9)
fsectorsphere	-32.2±1.02 (-32.3)	-35.6±1.71 (-35.)
fstepsphere	-15.9±13.0 (-25.3)	-16.1±12.7 (-25.3)
flnorm	-26.7±0.30 (-26.7)	-27.±0.26 (-27.2)
fbaluja	-4.27±0.46 (-4.19)	-4.37±0.28 (-4.50)
fschwefelrosen1	-35±0.90 (-35.)	-37.4±0.76 (-37.3)
fschwefelrosen2	-35.2±1.15 (-34.8)	-36.9±0.80 (-36.7)
fconcentric	-0.88±0.75 (-0.70)	-1.55±0.90 (-1.72)
fgriewank	-32.4±9.17 (-34.7)	-38.5±11.8 (-43.8)
frastrigin	1.66±0.65 (1.7)	1.47±0.68 (1.60)

Figure 4: Comparison between CMA and DCMA in dimension 2, 4 and 6. DCMA outperforms CMA in all experiments. The comparisons are not always significant, but most of them are, and the overall significance is strong. The overall significance can be shown e.g. by Hoeffding's one-sided inequality; as DCMA has a better average score than CMA for 42 experimental conditions on 42, Hoeffding's one-sided inequality applied to the average probability of a better score for DCMA than CMA over this 42 experimental conditions shows that it is larger than 1/2 with p-value $7.6e-10$.

5. CONCLUSION

We developed a theory of derandomization in optimization. Based on this theory, we proposed a simple and efficient improvement of CMA, a well-known evolution strategy. The improvement holds both in small dimension (the negative log of the fitness is multiplied by roughly 1.4 for almost all functions in dimension 2, 1.5 for $fgriewank$, and the (positive) log is very strongly divided for $fbaluja$) and in larger dimensions like 40 and 50.

An important point is that we claim that DCMA is better than CMA in all tested cases. The improvement depends on dimensions and functions, but all significant differences are in favor of DCMA. It is more efficient for almost all functions in all dimensions. Also, there's no drawback in terms of computational cost; the computation time of Halton numbers is very small, indeed usually smaller than the generation of classical pseudo-random numbers. Therefore, all improvements come for free.

The efficiency of DCMA is interpreted as a consequence of (i) better exploration, same effects make quasi-random-search better than random search; (ii) better exploitation, thanks to a better conditioning of the covariance-matrix-

Problem (dim8)	CMA	DCMA
fsphere	-34.5±0.60 (-34.7)	-37±0.77 (-37.4)
fsphereoneax	-36.±3.25 (-35.9)	-37.7±2.41 (-38.0)
frandsphere	-23.0±11.2 (-29.3)	-28.2±10.4 (-32.9)
fspherelb0	-34.±0.51 (-34.6)	-37.5±0.71 (-37.7)
fspherehull	-27.4±0.25 (-27.4)	-27.5±0.28 (-27.5)
fsectorsphere	-32.3±1.62 (-31.7)	-34.7±1.94 (-34.8)
fstepsphere	-16.1±12.7 (-25.3)	-15.8±13.1 (-25.3)
flnorm	-26.6±0.46 (-26.6)	-26.7±0.34 (-26.)
fbaluja	-3.93±0.26 (-3.98)	-4.02±0.31 (-3.96)
fschwefelrosen1	-34.0±0.81 (-34.2)	-36.8±0.67 (-36.8)
fschwefelrosen2	-34.6±0.63 (-34.8)	-36.8±0.41 (-36.9)
fconcentric	-0.74±0.82 (-0.70)	-0.65±0.89 (-0.50)
fgriewank	-33.±0.65 (-34.1)	-31.6±17.5 (-42.8)
frastrigin	2.18±0.51 (1.94)	1.97±0.57 (2.29)
Problem (dim10)	CMA	DCMA
fsphere	-34.8±0.64 (-34.7)	-36±0.73 (-36.9)
fsphereoneax	-36.8±3.44 (-36.5)	-37.0±2.19 (-36.8)
frandsphere	-10.1±4.96 (-9.21)	-11.0±7.24 (-9.98)
fspherelb0	-34.6±0.98 (-34.9)	-37.0±0.73 (-36.)
fspherehull	-27.0±0.24 (-27.1)	-27.2±0.25 (-27.2)
fsectorsphere	-28.5±1.4 (-28.8)	-33.3±1.18 (-33.4)
fstepsphere	-11.±13.5 (1e-11)	-18.3±11.9 (-25.3)
flnorm	-26.2±0.12 (-26.2)	-26.4±0.25 (-26.4)
fbaluja	-3.02±1.11 (-3.03)	-3.78±0.25 (-3.76)
Problem (dim20)	CMA	DCMA
fsphere	-34.6±0.67 (-34.7)	-36.1±0.27 (-36.0)
fsphereoneax	-15.8±2.64 (-16.6)	-36.8±1.92 (-36.9)
frandsphere	-0.40±0.74 (-0.36)	-1.71±1.3 (-2.)
fspherelb0	-34.3±0.62 (-34)	-36.1±0.37 (-36.1)
fspherehull	-20.2±0.74 (-20.2)	-25.5±0.50 (-25.5)
fsectorsphere	-10.6±1.28 (-11.0)	-15.1±1.53 (-15.4)
fstepsphere	-11.0±13. (1e-11)	-20.7±10.2 (-25.3)
flnorm	-17.0±0.92 (-17.)	-22.6±0.70 (-22.7)
fbaluja	11.5±0.00 (11.5)	11.5±8.51e-05 (11.5)
fschwefelrosen1	-34.±0.71 (-34.0)	-35.6±0.56 (-35.5)
fschwefelrosen2	-33.8±0.58 (-33.8)	-36.0±0.59 (-36.2)
fconcentric	0.87±0.22 (0.92)	0.75±0.12 (0.80)
fgriewank	-31.±8.85 (-33.9)	-32.±9.25 (-35.1)
frastrigin	3.43±0.13 (3.39)	3.05±0.31 (3.08)

Figure 5: Comparison between CMA and DCMA in dimension 8,10,20. DCMA outperforms CMA in all but three (non-significant) experiments in dimension 8 and all experiments in dimension 10, 20. Some comparisons are not significant but the overall significance is strong. Applying Hoeffding's one-sided test as in figure 4 for evaluating the overall significance leads to a p-value of $1.16e-05$.

Problem(dim30)	CMA	DCMA
fsphere	-27.9±1.06 (-27.)	-33.±0.91 (-34.1)
fsphereoneax	-15.9±4.59 (-15.6)	-25.9±2.19 (-25.6)
frandsphere	0.85±1.33 (0.82)	-0.30±0.95 (-0.65)
fspherelb0	-28.±1. (-28.5)	-34.4±0.53 (-34.5)
fspherehull	-13.7±0.42 (-13.6)	-16.7±0.49 (-16.8)
fsectorsphere	-3.82±0.69 (-3.97)	-7.72±1.07 (-7.54)
fstepsphere	-11±13.3 (1e-11)	-16.0±12.9 (-25.3)
flnorm	-10.3±0.87 (-10.5)	-14.1±0.40 (-14.0)
fbaluja	11.5±1.10e-06 (11.5)	11.5±1.35e-06 (11.5)
fschwefelrosen1	-20.5±0.92 (-20.7)	-26.6±0.86 (-26.6)
fschwefelrosen2	-20.1±0.92 (-20.)	-26.4±0.64 (-26.5)
fconcentric	1.06±0.13 (1.02)	1.01±0.07 (1.02)
fgriewank	-22.1±1.22 (-21)	-27.2±0.63 (-27.2)
frastrigin	3.84±0.29 (3.86)	3.68±0.32 (3.68)
Problem (dim40)	CMA	DCMA
fsphere	-21.8±0.96 (-21.9)	-25.6±0.50 (-25.7)
fsphereoneax	-14.±3.14 (-13.9)	-21.±3.46 (-21.1)
frandsphere	1.83±0.70 (1.78)	0.91±0.74 (1.12)
fspherelb0	-21.7±0.48 (-21.8)	-25.1±0.53 (-25.3)
fspherehull	-10.±0.19 (-10.5)	-12.4±0.38 (-12.6)
fsectorsphere	-0.88±0.66 (-0.71)	-3.52±0.41 (-3.54)
fstepsphere	-6.24±12. (1e-11)	-11.0±13.6 (1e-11)
flnorm	-7.5±0.57 (-7.49)	-9.72±0.43 (-9.73)
fbaluja	11.5±1.71e-07 (11.5)	11.5±1.96e-07 (11.5)
fschwefelrosen1	-14.2±0.65 (-14.1)	-18.0±0.76 (-17.8)
fschwefelrosen2	-14.2±0.67 (-14.3)	-17.2±0.53 (-17.3)
fconcentric	1.14±0.03 (1.12)	1.05±0.04 (1.02)
fgriewank	-15±3.43 (-15.8)	-18.5±0.69 (-18.)
frastrigin	4.35±0.61 (4.27)	4.20±0.16 (4.22)
Problem(dim50)	CMA	DCMA
fsphere	-17.4±0.54 (-17.5)	-20.9±0.50 (-20.9)
fsphereoneax	-13.9±4.61 (-13.6)	-18.5±2.25 (-18.4)
frandsphere	2.13±0.83 (1.87)	1.28±0.52 (1.28)
fspherelb0	-18.0±0.52 (-18.1)	-20.6±0.44 (-20.7)
fspherehull	-8.6±0.29 (-8.56)	-10.1±0.21 (-10.1)
fsectorsphere	0.69±0.88 (0.92)	-0.72±0.38 (-0.64)
fstepsphere	-6.1±12.3 (0.69)	-4.02±10.5 (1e-11)
flnorm	-5.34±0.62 (-5.38)	-6.87±1.39 (-7.30)
fbaluja	11.5±7.97e-08 (11.5)	11.5±5.65e-08 (11.5)
fschwefelrosen1	-10.1±0.79 (-10.2)	-12.5±0.70 (-12.3)
fschwefelrosen2	-10.1±0.75 (-10.1)	-12.3±0.43 (-12.3)
fconcentric	1.16±0.04 (1.12)	1.16±0.04 (1.12)
fgriewank	-11.5±2.37 (-12.)	-12.2±3.72 (-13.5)
frastrigin	4.8±0.50 (4.73)	4.5±0.24 (4.5)

Figure 6: Comparison between CMA and DCMA in dimension 30, 40, 50. DCMA outperforms CMA in all but one (non-significant, including ties) experiment in dimension 30 and 40 and all but three (non-significant, including ties) experiments in dimension 50. Some comparisons are not significant but the overall significance is strong. Applying Hoeffding's one-sided test as in figure 4 for evaluating the overall significance leads to an average probability of DCMA better than CMA over this 42 experimental conditions significantly larger than $1/2$ with p-value $5.1e-6$.

adaptation, as points are more properly distributed.

An important point is that these two advantages hold both in intra-offspring and inter-offsprings:

- The fact that RDCMA outperforms CMA shows that even *inside* a given offspring, derandomization is efficient.
- The fact that DCMA outperforms RDCMA shows that it is worth using random points which are "diversified" (more diversified than for random independent points) also *between* successive time steps.

Therefore we have naturally *two* forms of derandomization in DCMA. The same two derandomizations naturally occur for any $(1, \lambda)$ or $(1 + \lambda)$ evolution strategy, or any $(\mu/\mu, \lambda)$ or $(\mu/\mu+\lambda)$. For other evolution strategies, the second form of derandomization (between offsprings) is much harder.

The somewhat surprising fact that DdCMA is less efficient than CMA shows that avoiding step-size-redundancy between time steps might be damageable. This point, related to step-length adaptation, has to be further analyzed.

As a by-product of this study, we see the strong efficiency of scrambled quasi-random sequences in front of standard older quasi-random sequences, in particular when the dimension increases. We had very disappointing results with older, non-scrambled, quasi-random sequences.

Acknowledgements

We thank N. Hansen for kindly providing his implementation of CMAES in Octave/Matlab, the authors of GNU-Octave for having freely distributed Octave, and the authors of [4] for having freely provided their datasets on www. We thank A. Auger, C. Gagné, N. Hansen for fruitful talks.

6. REFERENCES

[1] A. Auger, M. Jebalia, and O. Teytaud. Xse: quasi-random mutations for evolution strategies. In *Proceedings of Evolutionary Algorithms, 12 pages*, 2005.

[2] A. Auger, M. Schoenauer, and O. Teytaud. Local and global order 3/2 convergence of a surrogate evolutionary algorithm. In *GECCO*, pages 857–864, 2005.

[3] H.-G. Beyer. *The Theory of Evolutions Strategies*. Springer, Heidelberg, 2001.

[4] P. Bratley and B. Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software Volume 14, Number 1, pages 88-100*, 1988.

[5] C. Cervellera and M. Muselli. A deterministic learning approach based on discrepancy. In *Proceedings of WIRN'03, pp53-60*, 2003.

[6] A. Conn, K. Scheinberg, and L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives, 1997.

[7] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. springer, 2003.

[8] S. Gelly, J. Mary, and O. Teytaud. On the ultimate convergence rates for isotropic algorithms and the best choices among various forms of isotropy. In 10^{th} *International Conference on Parallel Problem Solving from Nature (PPSN 2006)*, 2006.

[9] N. I. M. Gould, D. Orban, and P. L. Toint. Cuter and sifdec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, 2003.

[10] J. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[11] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 11(1), 2003.

[12] F. Hickernell. A generalized discrepancy and quadrature error bound. *Math. Comp. 67, 299-322*, 1998.

[13] S. Kimura and K. Matsumura. Genetic algorithms using low-discrepancy sequences. In *GECCO*, pages 1341–1346, 2005.

[14] P. L'Ecuyer and C. Lemieux. Recent advances in randomized quasi-monte carlo methods, 2002.

[15] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. 1992.

[16] A. Owen. *Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course*. 2003.

[17] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[18] P. Sarkar and M. Prasad. A comparative study of pseudo and quasi random sequences for the solution of integral equations. *J. Computational Physics, 68*, pages 66–88, 1978.

[19] I. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14(1):1–33, 1998.

[20] I. M. Sobol. On the systematic search in a hypercube. 16(5):790–793, Oct. 1979.

[21] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report AND KanGAL Report #2005005, IIT Kanpur, India, 2005.

[22] T.Back, F.Hoffmeister, and H.Schewefel. A survey of evolution strategies. Technical report, Dpt. of Computer Science XI, University of Dortmund, D-4600 , Dortmund 50, Germany, 1991.

[23] B. Tuffin. On the use of low discrepancy sequences in monte carlo methods. In *Technical Report 1060, I.R.I.S.A.*, 1996.