When does quasi-random work ?

Olivier Teytaud

When does quasi-random work ?

Olivier Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), Bât 490
Univ. Paris-Sud 91405 Orsay, France.
teytaud@lri.fr

Abstract. [10, 22] presented various ways for introducing quasi-random numbers or derandomization in evolution strategies, with in some cases some spectacular claims on the fact that the proposed technique was always and for all criteria better than standard mutations. We here focus on the quasi-random trick and see to which extent this technique is efficient, by an in-depth analysis including convergence rates, local minima, plateaus, non-asymptotic behavior and noise. We conclude to the very stable, efficient and straightforward applicability of quasi-random numbers in continuous evolutionary algorithms.

Keywords: Evolution Strategies; Derandomization.

1 Introduction

Whereas pseudo-random numbers are supposed to be as close as possible to pure random numbers, quasi-random numbers are designed in order to be more "uniformly" distributed than pure random numbers. Various criteria of uniformity have been developed [14, 9]. In some cases, a good value for a criterion ensures a good behavior: for example, a good discrepancy implies a small integration error for numerical integration of functions with finite total variation [8] through Koksma-Hlawka's inequality. Sequences with good properties for these criteria are mainly algebraic (quickly generated) methods [25, 14, 23, 16, 19, 3]. As a consequence of this important part of science, we can use fast and reliable generators of points, with good uniformity properties. Thanks to scrambling and other tricks (including randomization [11]) [2, 28, 5, 27, 13, 6, 15, 20, 29, 24, 1, 12, 17, 26], quasi-random points can be used also in high dimensionality with positive results [18].

Quasi-random numbers have been a revolution basically in numerical integration [14], but it was also used in other areas: quasi-random search [14], path planning [23], active learning [4], approximate dynamic programming [21].

[22], inspired by [14, 10], has proposed the use of quasi-random numbers for mutations in the continuous domain. The main results in [22] are about the convergence rate of CMA [7], modified by such an algorithm. However, many questions arised, about the generality of the results: as many people use random numbers as a secure tool for exploration, it is not intuitive that random numbers can be removed from evolutionary algorithms without strong drawbacks induced somewhere as a counterpart. We here provide an in-depth analysis of quasi-random mutations, through robustness, local minima, needle functions, noise, quality of the quasi-random number generator, and non-asymptotic properties. We conclude to the very wide applicability of quasi-random numbers in continuous evolution strategies.

In all the paper, we use the fitness functions presented in the Matlab/Octave implementation of CMA. When averages and standard deviations are presented, number between "(.)" are median values. Bold font indicate significant improvements, for rank-based tests (Wilcoxon statistics). The result of each algorithm is the best individual of the last generation.

The use of quasi-random mutations is quite straightforward. If your mutation operator is in Algorithm 1, then you just replace it by Algorithm 2.

The only difference is the replacement of $\mathcal{N}$, a standard multivariate Gaussian variable, by $\mathcal{N}_{\amalg\nabla}$, its quasi-random counterpart. If $\mathcal{N}$ is generated as in Algo. 3, then you just have to replace it by Algo. 4 in your favorite program.

In the rest of this paper, DCMA is the standard CMA (covariance matrix adaptation) algorithm, with the transformation above (from Algo. 3 to Algo. 4).

Algorithm 1 Standard mutation in the continuous domain, with σ a step size and A a linear transformation.

Function $x' = Mutation(x, \sigma, A)$
return $x' = x + \sigma A.\mathcal{N}$

Algorithm 2 Quasi-random mutation in the continuous domain, with σ a step size and A a linear transformation.

Function $x' = Mutation(x, \sigma, A)$
return $x' = x + \sigma A.\mathcal{N}_{qr}$

2 Quasi-random mutations need good quasi-random sequences

We compare a simple Halton sequence (without scrambling) and Sobol's sequence. It is known that Sobol sequence is much better, in particular for large dimensional problems, but the comparison is particularly impressive in the case of mutations of evolution strategies. Table 1 presents the comparison between the normalized (see caption) log of the smallest fitness value found by the algorithm in the case of Halton sequence compared to the random classical points and Sobol sequence for 10 function evaluations in dimension 4 and 40 function evaluations in dimension 16.

The explanation is clear on a typical plot of a random walk. Figure 1 (left) shows the sum of quasi-random Gaussian numbers generated with the 3rd and 4th variables of a Halton sequence. As well known for Halton's sequence (unscrambled version), we have long short term bias, leading to a quasi-random walk going away from 0. Figure 1 (right) shows also a quasi-random walk, generated with the partial sum of quasi-random Gaussian numbers generated with Sobol's sequence. The figure is very similar, visually, to what happens with a random walk.

It has already been pointed out in [22] that randomly rotating quasi-random Gaussian points at each offspring, in order to avoid some presumed bias, in useless and in fact reduces the efficiency of quasi-random points. We therefore here only use the standard version proposed above, without adding such rotation.

3 Quasi-random mutations improve the probability of finding a needle

It has been suggested that quasi-random points might be suitable only in regular cases, as they might be just more able of benefiting from artificial symetries in the problem. In order to test this assumption, we now consider the needle problem, defined as follows:

- the fitness value of any point at distance $\leqslant 1$ of $(1, \ldots, 1)$ is 0.
- the fitness value of any other point is $1 + R$, where R is a random independent uniform noise on $[0, 1]$.

The needle is found if at least one point of null fitness is found. We also defined the difficult needle problem, in which:

- the fitness value of any point at distance $\leqslant 1$ of $(K, K, \ldots, K)$ is 0.

Algorithm 3 Usual algorithm for generating a standard multivariate Gaussian.

Function $\mathcal{N} = MultivariateStandardGaussian(dimension\ d)$
for $i \in \{1, 2, \ldots, d\}$ **do**
  $x_i = random$ (uniform in $[0, 1]$)
end for
for $i \in \{1, 2, \ldots, d\}$ **do**
  $\mathcal{N}_i = inverseGaussianPDF(x_i)$
end for

Algorithm 4 Algorithm for generating quasi-random Gaussian numbers. Finally, only one simple loop is replaced by one call to a standard function.

Function $\mathcal{N} = MultivariateStandardGaussian(dimension\ d)$
Apply $x = Sobol(d)$.
for $i \in \{1, 2, \ldots, d\}$ **do**
  $\mathcal{N}_i = inverseGaussianPDF(x_i)$
end for

Problem	CMA	DCMA (Halton)	DCMA (Sobol)
10 function-evaluations in dimension 4			
fsphere	-0.169	*0.00688*	**-0.209**
fcigar	5.12	5.06	**4.97**
fstepsphere	-2.2	-1.58	**-2.84**
fconcentric	0.126	*0.15*	**0.0793**
fgriewank	-0.69	*-0.565*	**-0.952**
frastrigin	1.25	*1.29*	1.25
fschwefelmult	2.96	**2.96**	*2.96*
fsectorsphere	2.25	*4.05*	**1.95**
40 function-evaluations in dimension 16			
fsphere	0.648±0.0499	0.677±0.169	**0.547±0.0580**
fcigar	6.18±0.020	6.2±0.182	**6.14±0.0197**
fstepsphere	0.865±0.0330	0.79±0.146	**0.733±0.0585**
fconcentric	0.32±0.0108	*0.325±0.0405*	**0.297±0.0105**
fgriewank	-0.253±0.0365	-0.238±0.161	**-0.351±0.0353**
frastrigin	1.96±0.00719	1.98±0.0538	**1.94±0.00641**
fschwefelmult	**3.52±8.38e-05**	*3.52±0.000283*	*3.52±6.72e-05*
fsectorsphere	4.9±0.0912	4.96±1.17	4.93±0.0895

Table 1. $d \times \log(fitness)/n$ (i.e. the lower the better) for n function evaluations in dimension d, Comparison between standard CMA with random Gaussian numbers, and CMA with Sobol Gaussian numbers for very small numbers of iterations in dimension 4 and 16. Sobol points are equivalent to random points, whereas Halton points (without scrambling) lead to very poor results. Results in bold face are results in which a statistical difference with the random case appeared in the good direction (better than the random case); italic font is used for results in which a statistical difference occured in favor of usual random points: the difference is most often in favor of random points for Halton; and always in favor of Sobol except for the "fschwefelmult" function.

– the fitness value of any other point is $1 + R$, where R is a random independent uniform noise on $[0, 1]$.

As previously, the needle is found if at least one point of null fitness is found. The results are presented in table 2.

4 Quasi-random mutations improve the convergence rate

[22] has already strongly pointed out this fact, therefore we only briefly confirm these results in Figure 2. Except for "fschwefelmult", significant (95% confidence) results in favor of DCMA occur for all fitness functions for 2560 function-evaluations. We also present in table 3 the average log of fitness values for 10240 fitness evaluations in dimension 4 to see the asymptotic behavior for some fitness functions. Quasi-random points are better in all significant comparisons.

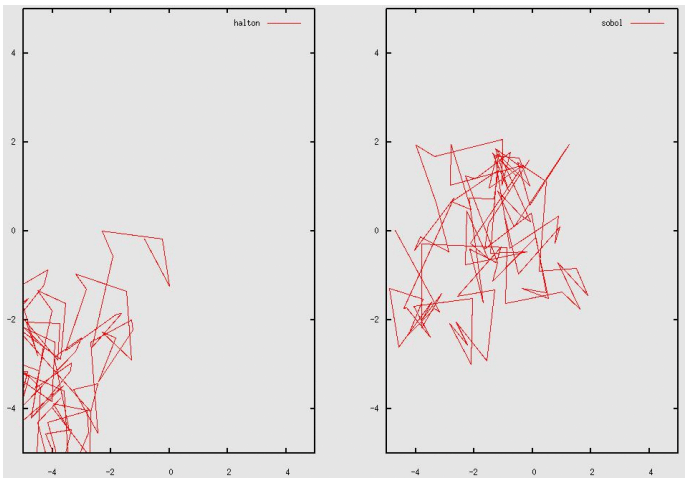

Fig. 1. 100 points of typical quasi-random walks with Gaussian quasi-random numbers generated with Halton (left) and Sobol (right). In the case of Halton, a strong short-term bias appears: the quasi-random walk goes away to the south-west.

Dimension	Number of fitness-evaluations	Probability of finding (CMA)	Probability of finding (DCMA)
3	64	36%± 3%	49 % ±3%
4	64	9 %± 2%	14 % ±2%
5	64	2.6 %± 1.3 %	7.3 % ± 2.1%

Standard needle

Dimension and K	Number of fitness-evaluations	Probability of finding (CMA)	Probability of finding (DCMA)
3, $K = 3$	64	0.8 % ± 0.4 %	2.4 % ± 0.6 %
4, $K = 2$	64	9.6 % ± 1.3 %	12.0 % ± 1.5%

Difficult needle

Table 2. Results of quasi-random points on the needle functions: probability of finding the needle (the higher the better). Quasi-random points work better.

5 Quasi-random mutations improve the non-asymptotic behavior: no log

Many papers consider the logarithm of the distance to the optimum, normalized by the dimension and/or the number of iterations, as the main criterion of quality of an optimization algorithm. The advantage of this approach is that the asymptotic behavior of continuous optimization algorithms, which is usually linear for evolution strategies, is clearly visible on such plots. However, the drawback of this approach is that the focus is on the convergence rate, and not on the probability of finding a good optimum.

In order to clearly point out the weakness of this criterion for multimodal optimization, let's consider the use of this criterion for evaluating a standard algorithm with known poor results for fitness functions with local minima.

Consider simply Newton's method with random initial point. The log of the distance ε_n (after n function evaluations) to the optimum is, for this algorithm, exponential as a function of n, if the initial point is sufficiently good. Consider p the probability of an initial point ensuring that the log-precision is at most k^{-2^n}, for some fixed $k > 1$. Then, the criterion $-\log(\varepsilon_n)/n$ has expectation at least $p \log(\varepsilon_n)/n = -p2^n \log(k)/n$. This criterion therefore tends to infinity for this Newton algorithm as $n \to \infty$, whenever the function is strongly non-convex and p is close to 0! We have therefore shown that this criterion will prefer an algorithm which converges with possibly very small probability, provided that n is sufficiently large, in front of any algorithm with linear convergence "only".

On the other hand, criteria like the expected fitness value certainly not have this behavior and show much more efficiently the probability of finding the optimum, in particular in the non-

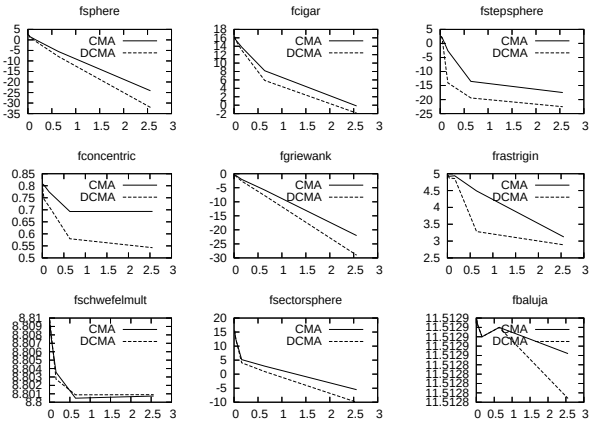

Fig. 2. Convergence rate of CMA and DCMA with Sobol in dimension 16. $Log(fitness)$ vs thousands of function evaluations. These objective functions are to be minimized. Standard deviations are not shown for the sake of readability; see text for significance.

Problem	CMA	DCMA
fsphere	-0.0140±0.000123 (-0.013)	**-0.0147±0.00014** (-0.0147)
fcigar	-0.0139±0.000150 (-0.0139)	**-0.0150±0.000125** (-0.0149)
fstepsphere	-0.00491±0.00128 (-0.00494)	**-0.00927±0.00061** (-0.00989)
fconcentric	-0.00088±3.99e-05 (-0.0009)	**-0.00109±3.65e-05** (-0.00113)
fgriewank	-0.0139±0.000175 (-0.0139)	**-0.0167±0.000174** (-0.0174)
frastrigin	8.41e-05±0.00018 (0.000427)	-0.000144±0.000222 (0.000427)
fschwefelmult	0.00289±1.20e-07 (0.00289)	0.00289±2.37e-08 (0.00289)
fsectorsphere	-0.0127±0.00012 (-0.0128)	**-0.0136±0.000147** (-0.0136)
fbaluja	-0.00187±3.88e-05 (-0.00186)	**-0.0019±3.15e-05** (-0.00194)

Table 3. Comparison of standard random numbers and quasi-random numbers from the point of view of the convergence rate. The numbers are the average log of fitness values (median between (.)); the quasi-random version is almost always significantly better (the lower the better).

asymptotic behavior. We therefore present below the average fitness value for fixed dimensions and numbers of fitness-evaluations. We point out that the same tables with the logarithm also lead to significant results in favor of DCMA - however, the results are more impressive with the expected fitness values as presented below.

The results in table 4 are the non-asymptotic counterpart of results in section 4 (which was convergence rate analysis). We here use 40 fitness-evaluations, for various dimensionality. [14] has already pointed out the non-asymptotic effect of quasi-random points in the simpler case of quasi-random search.

6 Quasi-random mutations deal efficiently with non-convex functions

Non-convex functions (multimodal functions, but also monomodal functions with plateaus and other non-convex functions like "fbaluja") are typically important fitness functions for which the convergence rate is moderately interesting: diversity loss in a plateau (or in an almost plateau) or premature convergence in a local minimum are a strong trouble. We here investigate the question of a possible loss of efficiency of quasi-random mutations for non-convex functions as a counterpart of positive effects pointed out in other sections of this paper. Table 5 show the average fitness value

Problem	CMA	DCMA
Dimension 2		
fsphere	0.0898±0.0356 (0.0172)	**0.0229±0.00762** (0.00436)
fcigar	21962.2±9678.1 (829.142)	**2415.2±726.955** (192.588)
fstepsphere	0.0660±0.0170 (3.36e-12)	0.0294±0.0142 (2.44e-12)
fconcentric	0.28±0.0501 (0.242)	**0.174±0.0432** (0.0972)
fgriewank	0.0282±0.0116 (0.00839)	**0.00620±0.00221** (0.0009)
frastrigin	2.94±0.237 (1.90)	**2.30±0.18** (1.51)
fschwefelmult	129.769±7.87 (100.287)	**108.386±7.04** (81.0)
fsectorsphere	2564.66±1540.54 (0.180)	0.491±0.0555 (0.175)
fbaluja	19661.4±2082.63 (18688.9)	**13146.7±1993.09** (8702.22)
Dimension 4		
fsphere	0.399±0.0412 (0.191)	**0.285±0.0425** (0.115)
fcigar	191315±34006.3 (83925.1)	**120477±18146.6** (49928.2)
fstepsphere	0.543±0.129 (0.311)	**0.188±0.0770** (0.0252)
fconcentric	0.243±0.0159 (0.177)	0.237±0.0143 (0.187)
fgriewank	0.069±0.0103 (0.0359)	**0.0408±0.00580** (0.0202)
frastrigin	5.25±0.413 (3.94)	4.65±0.292 (3.28)
fschwefelmult	240.185±28.52 (196.578)	**149.024±19.9** (134.868)
fsectorsphere	44113.8±24239.1 (2.25)	28279.3±12841.1 (1.23)
fbaluja	11035.8±652.625 (8418)	11828.5±741.584 (9322.88)
Dimension 16		
fsphere	1.93±0.138 (1.44)	1.7±0.120 (1.186)
fcigar	1.99e+06±143401 (1.39e+06)	1.72e+06±131583 (1.14e+06)
fstepsphere	1.93±0.133 (1.28)	1.75±0.112 (1.2)
fconcentric	0.310±0.0483 (0.199)	**0.199±0.0400** (0.134)
fgriewank	0.0689±0.00469 (0.0486)	0.0769±0.00493 (0.0570)
frastrigin	14.6±0.831 (11.5)	15.2±0.915 (11.6)
fschwefelmult	501.41±34.8 (351.154)	489.47±33.4 (324.699)
fsectorsphere	673136±57820.7 (414916)	595032±52901.3 (341269)
fbaluja	7774.31±507.891 (5136.37)	7646.29±483.262 (5719.69)

Table 4. Average fitness value after 40 function evaluations (the lower, the better).

(no log) of the best individual of the last offspring for various non-convex fitness functions. Very strong improvements sometimes appear with quasi-randomisation, and results were almost always significantly improved. All results below hold in dimension 4.

Problem	CMA	DCMA
10 function evaluations		
fstepsphere	1.5±0.146 (1)	**1.06±0.10** (1)
fconcentric	1.32±0.0261 (1.33)	**1.25±0.0259** (1.2)
fgriewank	0.209±0.0287 (0.143)	**0.138±0.0225** (0.100)
frastrigin	24.6±0.642 (24.5)	24.3±0.541 (24.1)
fschwefelmult	**1670.59±0.684** (1670.75)	1671.69±0.445 (1671.45)
fbaluja	99999.4±0.0267 (99999.4)	**99999.3±0.0348** (99999.4)
40 function evaluations		
Problem	CMA	DCMA
fstepsphere	1.5±0.32 (1)	**0.5±0.18** (1e-11)
fconcentric	1.41±0.110 (1.37)	**1.05 64±0.0981** (0.972)
fgriewank	0.188±0.0588 (0.0911)	**0.054±0.0138** (0.0308)
frastrigin	26.9±1.00 (27.0)	**23.9±0.876** (23.7)
fschwefelmult	1664.87±0.703 (1664.81)	**1662.15±0.378** (1661.48)
fbaluja	99999.2±0.108 (99999.3)	**99998.8±0.2** (99999.2)
160 function evaluations		
fstepsphere	0.4±0.128 (1e-11)	**0.125±0.0853** (1e-11)
fconcentric	0.7±0.096 (0.587)	**0.51±0.0623** (0.400)
fgriewank	0.0042±0.0015 (0.00237)	**0.000659±0.000481** (0.000153)
frastrigin	20.604±2.49 (21.5)	**6.83±1.74** (4.16)
fschwefelmult	1660.21±0.014 (1660.22)	**1660.16±0.00206** (1660.16)
fbaluja	99995.6±0.938 (99996.8)	**99986.3±2.86** (99991.8)
640 function evaluations		
fstepsphere	0.388±0.143 (1e-11)	**0.0555±0.0555** (1e-11)
fconcentric	0.232±0.0394 (0.128)	**0.120±0.0384** (0.0549)
fgriewank	0.000833±0.000411 (5.42e-11)	0.000245±0.000112 (1.51e-14)
frastrigin	5.57±0.427 (3.70)	**3.64±0.204** (2.98)
fschwefelmult	1660.15±5.98e-09 (1660.15)	**1660.15±6.90e-13** (1660.15)
fbaluja	94926.9±1036.12 (95873.5)	**13597.7±2714.8** (10272.5)

Table 5. Efficiency of quasi-random points for non-convex functions from a non-asymptotic point of view. The numbers are the average fitness values (the lower the better). For fschwefelmult, the significance holds but is hidden in late digits.

7 Quasi-random mutations deal efficiently with noise

Finally we tested fitness functions corrupted by noise. We just replaced the fitness function by its product with a random independent uniform number in $[0, 1]$, and we get the table 6 of results in dimension 2 with 160 function-evaluations.

In fact, the results are *more* impressive than the results of the non-noisy case: there's no decay of performance in the noisy case. We tested both the log and the no-log cases.

8 Conclusion

We have in this paper shown how general is the improvement induced by quasi-random numbers. In particular, quasi-random mutations lead to better results, not only from the point of view of the convergence rate , but also for several notions of robustness:

Problem	CMA	DCMA
fsphere	-0.12±0.00788	**-0.169±0.00469**
fcigar	0.013±0.0022	**-0.0205±0.00196**
fstepsphere	-0.341±0.00328	-0.344±0.00246
fconcentric	-0.0252±0.00096	**-0.030±0.00109**
fgriewank	-0.117±0.00490	**-0.167±0.0048**
frastrigin	-0.00444±0.00283	**-0.015±0.00413**
fschwefelmult	**0.053±0.00152**	0.0580±0.0014
fsectorsphere	-0.0391±0.0062	**-0.0829±0.00481**
fbaluja	**0.111±0.00142**	0.11±0.00141

$\log(fitness)$

Problem	CMA	DCMA
fsphere	0.000768±0.000372	**5.78e-06±2.55e-06**
fcigar	2819.93±2594.19	0.823±0.120
fstepsphere	0.0018±0.00138	0.00432±0.00422
fconcentric	0.207±0.0136	**0.163±0.0146**
fgriewank	0.00418±0.001	**0.000492±0.000297**
frastrigin	2.00±0.300	**1.20±0.224**
fschwefelmult	**111.731±10.198**	152.172±13.092
fsectorsphere	0.165±0.0641	**0.00304±0.000894**
fbaluja	**12775.2±1035.92**	17218±1348.25

$fitness$ value

Table 6. Mean $\log(fitness)$ and mean $fitness$ in the case of noise (see text for details). The lower, the better. Quasi-random numbers are almost always better.

- the improvement for the convergence rate scales from a few percents in the case of the sphere function or the cigar function to 80 % of speed-up in the case of the step-sphere function in dimension 4 - this suggests that is is much more the behavior in front of plateaus or needles which is improved, and not the behavior in a regular, smooth framework;
- the improvement remains when the fitness function is corrupted by noise; in fact, all comparisons are seemingly much more impressive when the fitness is corrupted by noise! Therefore, we claim that the effect ot quasi-random numbers are not limited to artificial smooth cases.
- the improvement is much more impressive on expected fitness values than on the convergence rate, because the quasi-random points improve the robustness and the probability of finding a solution as well as (and more than) the asymptotic convergence rate;
- in the case of small numbers of fitness values, one can also trust quasi-random points; the non-asymptotic behavior for non-logarithmic views on the fitness function (section 5) and small dimension provides impressive results: average fitness values are divided by factors between 3.9 (sphere function) and 9.1 (cigar function) in dimension 2. The results are less impressive in higher dimension (factor 1.16 in dimension 16 for the cigar function).
- quasi-random points are not afraid of non-convex fitness functions; we see 51% of improvement for the Griewank function with 10 function-evaluations in table 5, 3.2 for the step-sphere function and 53% for the Rastrigin function with 160 function-evaluations. In some other cases, the improvement is negligible, but we point out that it is never a loss of efficiency.
- the probability of finding a needle is improved also; the more difficult the needle problem, the higher the improvement; we conjecture that the improvement would be much higher with higher values of λ.

These results suggest that the improvement is due to (i) a better distribution of mutations inside one offspring but also in a cumulative manner over multiple steps as shown by earlier results in [22] (i.e., increasing independence between offsprings by random rotations of each offspring reduces the efficiency) (ii) less importantly, a better convergence rate by a better estimate of the position of the optimum - but this effect is probably much bigger for higher values of λ. We also tested the rate at which the covariance matrix is evaluated, but this effect is seemingly very small - perhaps this could be improved by better update rules. (i) includes/explains (a) the "init" effect is we consider as initialization the initial offsprings (b) the good "needle" effect (the step-sphere function, on which results are quite good, is typically a "multiple" needle problem, whereas (ii) explains the better convergence rate on the sphere function.

We also point out that replacing Monte-Carlo mutations by Quasi-Monte-Carlo points is straightforward: if your Gaussian points are generated thanks to the use of the reverse probability distribution function of the Gaussian on random uniform points, then just replace random independent vectors uniform on $[0, 1]$ by some quasi-random *good* generator. In particular, Sobol's sequence is very efficient and as fast as random points.

A further work is the design of step-size adaptation rules adapted to Quasi-Monte-Carlo mutations: the usual derivation of the cumulative step-size adaptation is not adapted with quasi-random mutations. However, results in this paper are quite stable and convincing without such adaptation; in all this paper we have used a standard cumulative step-size adaptation.

Acknowledgements The authors thank the many people who provided feedback on previous results around quasi-random for providing several of the questions which are answered in this article. We also thank A. Auger and N. Hansen for interesting feedback, in particular with respect to the future derivation of step-size adaptation rules for quasi-randomized mutations.

References

1. E.I. Atanassov. On the discrepancy of the halton sequences. *Math. Balkanica*, 18(12):1532, 2004.
2. E. Braaten and G. Weller. An improved low-discrepancy sequence for multidimensional quasi-monte carlo integration. *J. Comput. Phys.*, 33:249–258, 1979.
3. Paul Bratley and Bennett Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software Volume 14, Number 1, pages 88-100*, 1988.
4. Cristiano Cervellera and Marco Muselli. A deterministic learning approach based on discrepancy. In *Proceedings of WIRN'03, pp53-60*, 2003.
5. R. Cranley and T.N.L. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM J. Numer. Anal.*, 13(6):904914, 1976.
6. H. Faure. Good permutations for extreme discrepancy. *J. Number Theory*, 42:47–56, 1992.
7. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 11(1), 2003.
8. G.H. Hardy. On double fourier series, and especially those which represent the double zeta-function with real and incommensurable parameters. *Quart. J. Mathematics*, 37:53–79, 1905.
9. F.J. Hickernell. A generalized discrepancy and quadrature error bound. *Math. Comp. 67, 299-322*, 1998.
10. Shuhei Kimura and Koki Matsumura. Genetic algorithms using low-discrepancy sequences. In *GECCO*, pages 1341–1346, 2005.
11. P. L'Ecuyer and C. Lemieux. *Recent Advances in Randomized Quasi-Monte Carlo Methods*, pages 419 – 474. Kluwer Academic, 2002.
12. M. Mascagni and H. Chi. On the scrambled halton sequence. *Monte Carlo Methods Appl.*, 10(3):435–442, 2004.
13. W.J. Morokoff and R.E. Caflish. Quasi-random sequences and their discrepancies. *SIAM J. Sci. Comput.*, 15(6):12511279, 1994.
14. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. 1992.
15. G. Okten and A. Srinivasan. Parallel quasi-monte carlo methods on a heterogeneous cluster. In *in: H. Niederreiter, K.-T. Fang, F.J. Hickernell (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2000, Springer, Berlin, Heidelberg*, page 406421, 2002.
16. A.B. Owen. *Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course*. 2003.
17. P.K. Sarkar and M.A. Prasad. A comparative study of pseudo and quasi random sequences for the solution of integral equations. *J. Computational Physics, 68*, pages 66–88, 1978.
18. I.H. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14(1):1–33, 1998.
19. I. M. Sobol. On the systematic search in a hypercube. *Siam journal on Numerical Analysis*, 16(5):790–793, October 1979.
20. A. Srinivasan. Parallel and distributed computing issues in pricing financial derivatives through quasi-monte carlo. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, 2002.
21. O. Teytaud, S. Gelly, and J. Mary. Active learning in regression, with application to stochastic dynamic programming. In *Proceedings of ICINCO'07*, pages 47–54, 2007.
22. Olivier Teytaud and Sylvain Gelly. Dcma: yet another derandomization in covariance-matrix-adaptation. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 955–963, New York, NY, USA, 2007. ACM.
23. B. Tuffin. On the use of low discrepancy sequences in monte carlo methods, 1996.

24. B. Tuffin. A new permutation choice in halton sequences. *Monte Carlo and Quasi-Monte Carlo*, 127:427435, 1997.
25. J. G. van der Corput. Verteilungsfunktionen. *Proc. Ned. Akad. v. Wet., 38:813?821*, 1935.
26. B. Vandewoestyne and R. Cools. Good permutations for deterministic scrambled halton sequences in terms of l2-discrepancy. *Computational and Applied Mathematics*, 189(1,2):341:361, 2006.
27. X. Wang and F. Hickernell. Randomized halton sequences. *Math. Comput. Modelling*, 32:887–899, 2000.
28. T.T. Warnock. Computational investigations of low-discrepancy point sets. In *In: S.K. Zaremba, Editor, Applications of Number Theory to Numerical Analysis (Proceedings of the Symposium), University of Montreal*, page 319343, 1972.
29. T.T. Warnock. Computational investigations of low-discrepancy point sets ii. In *In: H. Niederreiter and P.J.-S. Shiue, Editors, Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer, Berlin*, 1995.