

DCMA, yet another derandomization in Covariance-Matrix-Adaptation - DRAFT

Olivier Teytaud, Sylvain Gelly

Equipe TAO - INRIA Futurs, LRI, Bat. 490, University Paris-Sud, 91405 Orsay
Cedex. France

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 restricted amount of random seeds, and apply this 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, computationally faster and the convergence is faster.

1 Introduction

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

- they are derivative-free and in most cases only depending on comparisons between fitness values (as well as direct search methods, see e.g. [6]);
- they usually work in numerical optimisation 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 that 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 is a well known evolution strategy. The reader is referred to [11] for details, but very roughly, its principle is as follows for each epoch:

1. randomly generate a population of p points according to the current parameters of the Gaussian;
2. depending on the quality of these points, update the parameters of the Gaussian so that it gets closer to the set of best points in the population;
3. go back to step 1.

In this paper, we improve two important qualities respectively from CMA-ES and ES in general by the use of quasi-random points. First, the main advantage of CMA-ES in front of other evolution strategies is the derandomization

of the direction of search, by the estimation of an ellipsoid containing the best points. However, it is known that the conditioning of 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 also among multiple time-steps, and not only inside a given offspring. This is handled by quasi-random points, as the offspring at one epoch will be "not-too-redundant" with the previous offsprings. This is the first advantage of our derandomization below.

Second, Evolution Strategies (ES), 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.

Quasi-random experiments in evolutionary computation have already been performed in [13] ; we here use (i) a more sophisticated evolutionary computation algorithm, (ii) a low-discrepancy sequence more suitable for high-dimension thanks to scrambling, (iii) a relation between quasi-random successive offsprings that is particularly easy thanks to the fact that only one gaussian is used to generate the population - as CMA-ES works on the mean/step-size/covariance of only one Gaussian distribution, we can derandomize a sequence of $n > \lambda$ mutations simply using the Gaussian version of the low-discrepancy sequence, whereas if the algorithm was working with many gaussians simultaneously (e.g. one per individual) it would be very hard to ensure some good property of non-redundancy of individuals with different parents. This will be detailed in the sequel; mainly, our method is easy to use in any method that uses one and only one distribution at each offspring, typically $(\mu/\mu, \lambda)$ -evolution strategies (with or without weighting of averages). Extensions are probably possible, but we will focus on this simpler case for this paper. $(\mu/\mu, \lambda)$ -algorithms in general, and CMA-ES in particular, are very efficient in many cases, in particular on the Cec'05 benchmark[21], and therefore their improvement is important.

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]), 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 derandomized algorithm is termed DCMA and uses quasi-random Gaussian mutations instead of random Gaussian mutations as in CMA. We also considered 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 is *grand* \times *rand*, where *grand* is a quasi-random unidimensional Gaussian number, coming from

the Van Der Corput sequence, with prime number 2, and *rand* is a usual unit random vector in the sphere. The paper is organized as follows. Section 2 presents some theoretical insights about randomness use. 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 roughly show that (i) randomization is necessary to ensure 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 unsmooth creation of many real numbers from this vector). The positive result with one random uniform number is reached thanks to a standard quasi-random sequence with a simple classical randomization.

An optimization algorithm is defined in the formalism of oracle as described in figure 1.

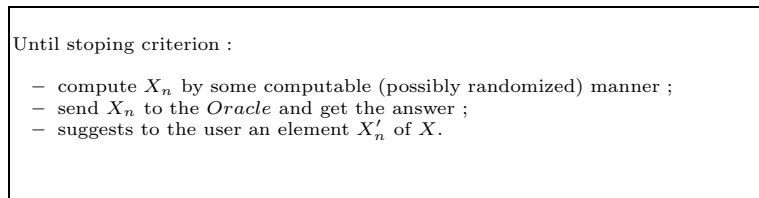


Fig. 1. Formalization of optimization algorithms.

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 1 (Optimization does not work in general). *We consider X infinite and non-countable (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 non-countable 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 $X_n^1 = u$. 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 non-countable.

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 non-countable. 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 set 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 $n \geq 1$, $\sum_w Q_{n,w} = 1 < \infty$. 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 non-countable. 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. \square

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 (Randomized optimization can find essential minima). Consider $X = [0, 1]^d$. Then, random search as follows:

- X_n is 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.

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

. Moreover,

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

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 3 (Finding the essential supremum requires a non-countable randomization). 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: Essentially, the proof follows the lines of the proof of theorem 1.

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

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

the set of all possibly visited 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. \square

Therefore, we have shown

- that finding the minimum on a continuous domain is too hard as a goal (theorem 1);
- that finding the essential minimum is possible by simple random search (theorem 2);
- that finding the essential minimum is not possible without at least a non-countable randomization (theorem 3). One could indeed refine theorem 3 by extending it to algorithms that have a non-countable 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).

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 search¹. 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 4 (One random point is enough for finding essential minima). *Consider the following quasi-random-search algorithm:*

- uniformly randomly generate $r \in X = [0, 1]^d$;
- consider x_1, x_2, \dots a (deterministic) sequence² 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.

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

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

$$\text{almost surely, } r + S \text{ intersects } Q_{e_n} \tag{1}$$

then 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 1 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, there exists Z of null measure such that $Q \cup Z$ contains an open ball B .³

B is open and S is dense; therefore $B - S = X$.

This means that

$$\begin{aligned} X &\subset \{b - s; b \in B, s \in S\} \\ X &\subset \{q - s; q \in Q, s \in S\} \cup \{z - s; z \in Z, s \in S\} \\ X &\subset (Q - S) \cup \underbrace{\{z - s; z \in Z, s \in S\}}_{\text{null Lebesgue measure}} \end{aligned}$$

$$\mu(X) \leq \mu(Q - S)$$

¹ Note however that interesting questions can be studied about the computability of such an approach in various models of computation on real numbers.

² In particular, quasi-random numbers can be used.

³ We recall that for any Lebesgue-measurable set A , there exists a borel-set B with same measure such that the symmetric-difference between A and B has null Lebesgue-measure.

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. \square

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. Essentially, such an analysis has already been published in the case of quasi-random-search ([15]).

We now move to concrete derandomization. As we will see in the sequel with a moderate yet significant derandomization that strongly improves empirical results, the statement that derandomization is possible, but “not too much” derandomization, makes sense in practice also.

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]).

So, there are “good” point sets and “bad” point sets. In order to generate good point sets, a measure of goodness would be useful. Consider a point set x_1, \dots, x_n in $D = [0, 1]^d$. An intuitive idea is $\sup_{x \in D} \inf_{i \in \{1, \dots, n\}} d(x, x_i)$ (to be minimized), where d is some distance (e.g. L^∞ distance). Then, generating points as the lattice in figure 2 is easy, and it has been pointed out that for the criterion above, this is optimal for many values of the number of points ([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 a good point set. This is not the case for the lattice in figure 2 : the projection on some well-chosen axis leads to accumulations. Therefore, other criterions have been defined. The most well known criterion is discrepancy. There are various discrepancies. The most well known is the following : $\sup_{r \in D} \left| \frac{1}{n} \text{Card}\{i \in \{1, \dots, n\}; \forall j (x_i)_j \leq r_j\} - \pi_{j \in \{1, \dots, d\}} r_j \right|$. 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, \dots, x_n is not the discrepancy of $1 - x_1, \dots, 1 - x_n$;
- it is a worst case on r .

The two first elements can be debated. Considering variables, in a non-rotation invariant manner, 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 :

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

This form of discrepancy (as well as others) verify 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]).

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 : $n = p_k p_{k-1} \dots p_1$, i.e. $n = \sum_{i=1}^k p_i p^i$ with $p_i \in [[0, p-1]]$;
- $x_{n,p} = 0.p_1 p_2 \dots p_k$ in basis p .

A classical improvement, termed *scrambling* consists in replacing this by $x_{n,p} = 0.\pi(p_1)\pi(p_2)\dots\pi(p_k)$ where π is some permutation of $0, 1, \dots, p-1$ 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

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

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) :

- 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 set $n \leftarrow n + p$ (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).

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

4 Experimental comparisons

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, and within parenthesis the median, for various classical test functions. All results come from 11 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. It is not clearly different from scrambled-Halton.

RDCMA is 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.

5 Conclusion

We developed a short theory of derandomization in optimization. Based on this theory, we proposed a simple and efficient improvement of CMA, a well-known evolution strategies. The results are much better in small dimension, (the log 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 even in dimension 40 and 50 the improvement is significant and can be very impressive for some functions that require a strong exploration.

An important point is that we claim that DCMA is *uniformly* better than CMA. It is more efficient for almost all functions in all dimensions. Also, there's no drawback in terms of computational overhead; the computation time of Halton numbers is very small, indeed usually smaller than calls to classical pseudo-random numbers. Therefore, all improvements come for free.

The efficiency of DCMA is interpreted as a consequence of (i) better exploration, as well as quasi-random-search is better than random search ; (ii) better conditioning of the covariance-matrix-adaptation, as points are more properly distributed.

An important point is that this two advantages holds both 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 building a directional exploration which is "diversified" (more diversified than for random points) *between* successive time steps. Therefore we have naturally *two* forms of derandomization in DCMA. The same two derandomization 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. In particular, non-presented results with deterministic sequences without scrambling are very disappointing when dimension increases.

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.

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 *10th 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 Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog 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. Schwefel. 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, 1996.

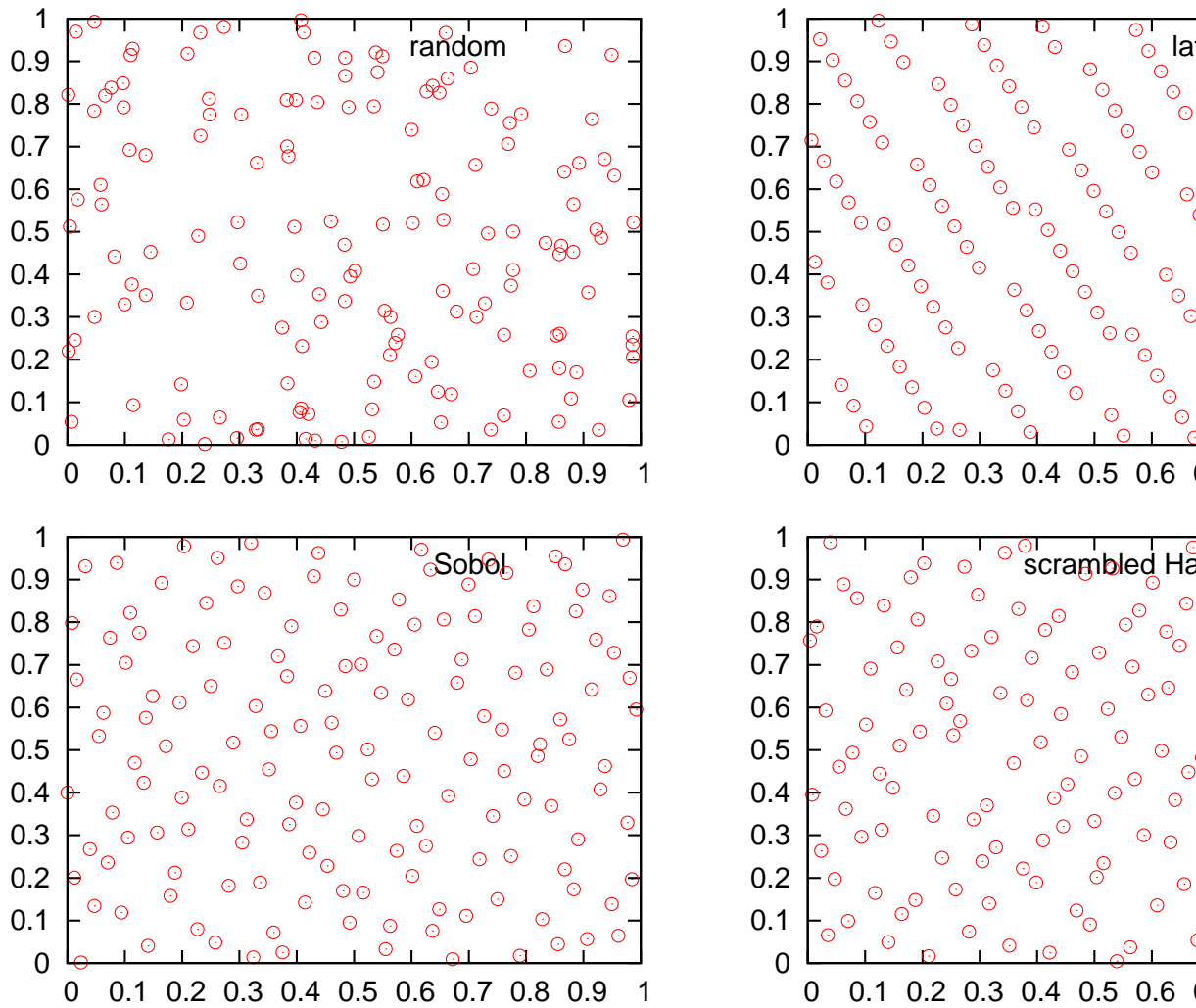


Fig. 2. Random points, lattice points, Sobol points, scrambled-Halton points.

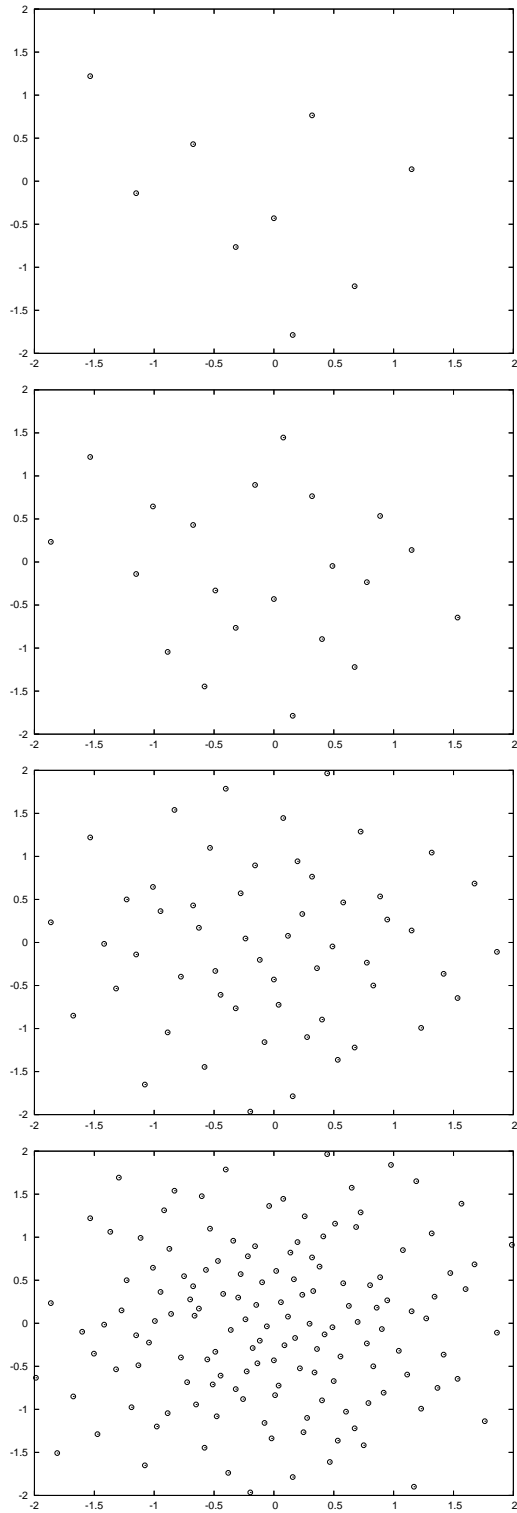


Fig. 3. 9, 21, 57 and 153 quasi-random gaussian points in dimension 2. See text for details.

Problem (dim2)	CMA	DCMA	DCMA-Sobol
fsphere	-28.2±2.52 (-28.5)	-41±1.98 (-41.)	-40.6±0.912 (-40.4)
fsphereoneax	-30.0±5.57 (-30.5)	-41.7±2.87 (-42.0)	-42.±3.16(-41.6)
frandsphere	-28.2±2.84 (-27.2)	-38.7±0.946 (-38.8)	-39.±1.77(-39.0)
fsphere1b0	-31.2±2.21 (-31.)	-42.±1.36 (-42.3)	-40.8±1.67(-40.5)
fspherehull	-15.0±1.83 (-14.3)	-21.2±1.00 (-21.4)	-20.8± 1.22 (-20.8)
fsectorsphere	-15.0±2.24 (-15.9)	-21.0±1.80 (-21.5)	-21.9± 1.24 (-21.9)
fstepsphere	-11.5±13.2 (1e-11)	-20.7±10.2 (-25.3)	-20.7 ± 10.2 (-25.3)
fnorm	-13.5±1.82 (-14.2)	-20±0.924 (-21.2)	-20.2±0.823 (-19.8)
fbaluja	7.92±1.17 (8.39)	1.19±1.37 (1.21)	2.11±0.72(1.91)
fschwefelrosen1	-26.9±2.95 (-26.4)	-40.2±1.87 (-40.)	-38.6±1.55 (-39.)
fschwefelrosen2	-27.0±4.23 (-26.3)	-38.5±0.773 (-38.8)	-39.2± 1.40 (-39.4)
fconcentric	-3.51±1.11 (-4.14)	-5.70±1.16 (-5.22)	-5.60±1.74 (-5.22)
fgriewank	-32.2±2.75 (-32.9)	-48.8±1.83 (-48.)	-48.0 ±1.21 (-48.1)
frastrigin	0.322±0.377 (-0.00504)	-11.6±16.9 (-0.00505)	-6.21± 14.6 (-0.00505)
Problem (dim4)	CMA	DCMA	DCMA-Sobol
fsphere	-35.7±1.50 (-35.3)	-38.5±1.35 (-38.5)	-37.3±1.03 (-37.7)
fsphereoneax	-35.4±2.46 (-34.8)	-39.8±2.61 (-40.0)	-41.8±3.06 (-42.3)
frandsphere	-33.9±2.04 (-33.2)	-36.0±1.02 (-35.8)	-35.7±2.14 (-35.6)
fsphere1b0	-36.4±1.24 (-35.8)	-38.2±1.12 (-37.)	-37.6±1.11 (-37.2)
fspherehull	-27.1±0.967 (-27.4)	-28.1±0.425 (-28.0)	-28.±0.403(-28.0)
fsectorsphere	-25.9±3.34 (-26.2)	-34.5±1.68 (-34.1)	-34.±1.49 (-35.4)
fstepsphere	-23.0±7.63 (-25.3)	-25.3±3.72e-15 (-25.3)	-13.7±13.3(-25.3)
fnorm	-26.1±1.09 (-26.5)	-27.6±0.453 (-27.5)	-27.9±0.586 (-27.)
fbaluja	-1.39±1.14 (-1.42)	-5.21±0.47 (-5.11)	-4.99±0.504(-4.93)
fschwefelrosen1	-35.3±0.757 (-35.4)	-38.6±0.609 (-38.6)	-37.±1.00 (-37.2)
fschwefelrosen2	-36.0±1.89 (-36.2)	-37.±1.15 (-37.9)	-38.2±0.803 (-38.0)
fconcentric	-2.41±1.23 (-2.90)	-3.24±1.07 (-3.46)	-2.65±1.27 (-2.90)
fgriewank	-35.6±2.92 (-35.1)	-42.0±12.4 (-45.4)	-43.9±3.66 (-44.9)
frastrigin	1.30±0.863 (1.38)	0.849±0.688 (0.688)	0.456±0.600 (-0.00505)
Problem (dim6)	CMA	DCMA	DCMA-Sobol
fsphere	-34.±0.854 (-34.9)	-37.±1.06 (-37.8)	-35.±1.17 (-35.4)
fsphereoneax	-35.7±2.16 (-35.0)	-37.2±2.8 (-37.2)	-35.9±1.84 (-35.5)
frandsphere	-29.6±6.14 (-31.2)	-34.5±0.930 (-34.7)	-31.9±5.79 (-33.7)
fsphere1b0	-34.8±0.622 (-34.8)	-37.8±0.676 (-37.6)	-35.7 ±0.412 (-35.6)
fspherehull	-27.5±0.339 (-27.4)	-27.9±0.225 (-27.9)	-27.7± 0.506 (-27.)
fsectorsphere	-32.2±1.02 (-32.3)	-35.6±1.71 (-35.)	-34.5±2.16 (-34.1)
fstepsphere	-15.9±13.0 (-25.3)	-16.1±12.7 (-25.3)	-20.4 ± 10.8 (-25.3)
fnorm	-26.7±0.301 (-26.7)	-27.±0.265 (-27.2)	-27.0±0.227 (-27.0)
fbaluja	-4.27±0.46 (-4.19)	-4.37±0.282 (-4.50)	-4.46±0.147 (-4.45)
fschwefelrosen1	-35±0.908 (-35.)	-37.4±0.766 (-37.3)	-36.±0.996 (-36.9)
fschwefelrosen2	-35.2±1.15 (-34.8)	-36.9±0.807 (-36.7)	-36.5±1.16 (-36.1)
fconcentric	-0.882±0.752 (-0.707)	-1.55±0.902 (-1.72)	-1.63±0.825 (-2.05)
fgriewank	-32.4±9.17 (-34.7)	-38.5±11.8 (-43.8)	-30.5±12.9 (-35.3)
frastrigin	1.66±0.652 (1.7)	1.47±0.680 (1.60)	1.71±0.394 (1.60)

Fig. 4. Comparison between CMA and DCMA in dimension 2, 4 and 6. DCMA strongly outperforms CMA in all experiments. The comparisons are not always significant, but most of them are, and the overall significance is strong. The improvement over CMA clearly decreases as dimension increases.

Problem (dim8)	CMA	DCMA	DCMA-Sobol
fsphere	-34.5±0.605 (-34.7)	-37±0.773 (-37.4)	-36.7±0.423 (-36.7)
fsphereoneax	-36.±3.25 (-35.9)	-37.7±2.41 (-38.0)	-37.2±2.5 (-37.7)
frandsphere	-23.0±11.2 (-29.3)	-28.2±10.4 (-32.9)	-23.±9.08(-23.3)
fsphereIb0	-34.±0.511 (-34.6)	-37.5±0.717 (-37.7)	-36.2±0.527 (-36.2)
fspherehull	-27.4±0.256 (-27.4)	-27.5±0.281 (-27.5)	-27.5±0.251 (-27.6)
fsectorsphere	-32.3±1.62 (-31.7)	-34.7±1.94 (-34.8)	-34.2±1.70 (-27.6)
fstepsphere	-16.1±12.7 (-25.3)	-15.8±13.1 (-25.3)	-15.±13.3(-25.3)
fnorm	-26.6±0.463 (-26.6)	-26.7±0.347 (-26.)	-26.8±0.315(-26.7)
fbaluja	-3.93±0.26 (-3.98)	-4.02±0.318 (-3.96)	-3.98±0.216(-3.9)
fschwefelrosen1	-34.0±0.814 (-34.2)	-36.8±0.676 (-36.8)	-36.6±0.646 (-36.5)
fschwefelrosen2	-34.6±0.631 (-34.8)	-36.8±0.417 (-36.9)	-36.7±0.683 (-36.9)
fconcentric	-0.740±0.821 (-0.707)	-0.659±0.896 (-0.506)	-0.478± 0.780 (-0.707)
fgriewank	-33.±0.659 (-34.1)	-31.6±17.5 (-42.8)	-38.7±3.27 (-36.7)
frastrigin	2.18±0.516 (1.94)	1.97±0.576 (2.29)	1.96±0.434 (1.94)
Problem (dim10)	CMA	DCMA	DCMA-Sobol
fsphere	-34.8±0.641 (-34.7)	-36±0.735 (-36.9)	-36.±0.912 (-36.7)
fsphereoneax	-36.8±3.44 (-36.5)	-37.0±2.19 (-36.8)	-36.6±2.44 (-35.8)
frandsphere	-10.1±4.96 (-9.21)	-11.0±7.24 (-9.98)	-17.3±9.24 (-14.2)
fsphereIb0	-34.6±0.988 (-34.9)	-37.0±0.735 (-36.)	-35.8±0.63 (-35.9)
fspherehull	-27.0±0.248 (-27.1)	-27.2±0.256 (-27.2)	-27.2±0.173 (-27.2)
fsectorsphere	-28.5±1.4 (-28.8)	-33.3±1.18 (-33.4)	-33.8±1.44 (-34.1)
fstepsphere	-11.±13.5 (1e-11)	-18.3±11.9 (-25.3)	-18.2±12.0(-25.3)
fnorm	-26.2±0.129 (-26.2)	-26.4±0.255 (-26.4)	-26.2±0.253 (-26.2)
fbaluja	-3.02±1.11 (-3.03)	-3.78±0.250 (-3.76)	-3.4±0.37 (-3.57)
Problem (dim20)	CMA	DCMA	
fsphere	-34.6±0.677 (-34.7)	-36.1±0.272 (-36.0)	
fsphereoneax	-15.8±2.64 (-16.6)	-36.8±1.92 (-36.9)	
frandsphere	-0.403±0.743 (-0.36)	-1.71±1.3 (-2.)	
fsphereIb0	-34.3±0.622 (-34)	-36.1±0.378 (-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)	
fnorm	-17.0±0.921 (-17.)	-22.6±0.700 (-22.7)	
fbaluja	11.5±0.00114 (11.5)	11.5±8.51e-05 (11.5)	
fschwefelrosen1	-34.±0.71 (-34.0)	-35.6±0.569 (-35.5)	
fschwefelrosen2	-33.8±0.58 (-33.8)	-36.0±0.598 (-36.2)	
fconcentric	0.871±0.223 (0.920)	0.754±0.120 (0.809)	
fgriewank	-31.±8.85 (-33.9)	-32.±9.25 (-35.1)	
frastrigin	3.43±0.13 (3.39)	3.05±0.314 (3.08)	

Fig. 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.

Problem(dim30)	CMA	DCMA
fsphere	-27.9±1.06 (-27.)	-33.±0.914 (-34.1)
fsphereoneax	-15.9±4.59 (-15.6)	-25.9±2.19 (-25.6)
frandsphere	0.852±1.33 (0.823)	-0.301±0.959 (-0.655)
fspherelb0	-28.±1. (-28.5)	-34.4±0.539 (-34.5)
fspherehull	-13.7±0.429 (-13.6)	-16.7±0.491 (-16.8)
fsectorsphere	-3.82±0.693 (-3.97)	-7.72±1.07 (-7.54)
fstepsphere	-11±13.3 (1e-11)	-16.0±12.9 (-25.3)
fnorm	-10.3±0.875 (-10.5)	-14.1±0.408 (-14.0)
fbaluja	11.5±1.10e-06 (11.5)	11.5±1.35e-06 (11.5)
fschwefelrosen1	-20.5±0.920 (-20.7)	-26.6±0.861 (-26.6)
fschwefelrosen2	-20.1±0.929 (-20.)	-26.4±0.643 (-26.5)
fconcentric	1.06±0.138 (1.02)	1.01±0.0732 (1.02)
fgriewank	-22.1±1.22 (-21)	-27.2±0.638 (-27.2)
frastrigin	3.84±0.291 (3.86)	3.68±0.325 (3.68)
Problem (dim40)	CMA	DCMA
fsphere	-21.8±0.966 (-21.9)	-25.6±0.509 (-25.7)
fsphereoneax	-14.±3.14 (-13.9)	-21.±3.46 (-21.1)
frandsphere	1.83±0.705 (1.78)	0.916±0.747 (1.12)
fspherelb0	-21.7±0.482 (-21.8)	-25.1±0.538 (-25.3)
fspherehull	-10.±0.191 (-10.5)	-12.4±0.380 (-12.6)
fsectorsphere	-0.88±0.664 (-0.71)	-3.52±0.414 (-3.54)
fstepsphere	-6.24±12. (1e-11)	-11.0±13.6 (1e-11)
fnorm	-7.5±0.571 (-7.49)	-9.72±0.431 (-9.73)
fbaluja	11.5±1.71e-07 (11.5)	11.5±1.96e-07 (11.5)
fschwefelrosen1	-14.2±0.658 (-14.1)	-18.0±0.762 (-17.8)
fschwefelrosen2	-14.2±0.678 (-14.3)	-17.2±0.533 (-17.3)
fconcentric	1.14±0.0398 (1.12)	1.05±0.047 (1.02)
fgriewank	-15±3.43 (-15.8)	-18.5±0.697 (-18.)
frastrigin	4.35±0.614 (4.27)	4.20±0.165 (4.22)
Problem(dim50)	CMA	DCMA
fsphere	-17.4±0.546 (-17.5)	-20.9±0.500 (-20.9)
fsphereoneax	-13.9±4.61 (-13.6)	-18.5±2.25 (-18.4)
frandsphere	2.13±0.832 (1.87)	1.28±0.529 (1.28)
fspherelb0	-18.0±0.523 (-18.1)	-20.6±0.449 (-20.7)
fspherehull	-8.6±0.29 (-8.56)	-10.1±0.213 (-10.1)
fsectorsphere	0.691±0.888 (0.926)	-0.722±0.383 (-0.648)
fstepsphere	-6.1±12.3 (0.693)	-4.02±10.5 (1e-11)
fnorm	-5.34±0.628 (-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±0.796 (-10.2)	-12.5±0.702 (-12.3)
fschwefelrosen2	-10.1±0.750 (-10.1)	-12.3±0.430 (-12.3)
fconcentric	1.16±0.0487 (1.12)	1.16±0.0494 (1.12)
fgriewank	-11.5±2.37 (-12.)	-12.2±3.72 (-13.5)
frastrigin	4.8±0.504 (4.73)	4.5±0.240 (4.5)

Fig. 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.