A Rigorous Runtime Analysis for Quasi-Random Restarts and Decreasing Stepsize

Marc Schoenauer, Fabien Teytaud, Olivier Teytaud

A Rigorous Runtime Analysis for Quasi-Random Restarts and Decreasing Stepsize

Marc Schoenauer, Fabien Teytaud and Olivier Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), bat 490 Univ. Paris-Sud
91405 Orsay, France

Abstract. Multi-Modal Optimization (MMO) is ubiquitous in engineering, machine learning and artificial intelligence applications. Many algorithms have been proposed for multimodal optimization, and many of them are based on restart strategies. However, only few works address the issue of initialization in restarts. Furthermore, very few comparisons have been done, between different MMO algorithms, and against simple baseline methods. This paper proposes an analysis of restart strategies, and provides a restart strategy for any local search algorithm for which theoretical guarantees are derived. This restart strategy is to decrease some 'step-size', rather than to increase the population size, and it uses quasi-random initialization, that leads to a rigorous proof of improvement with respect to random restarts or restarts with constant initial step-size. Furthermore, when this strategy encapsulates a (1+1)-ES with 1/5th adaptation rule, the resulting algorithm outperforms state of the art MMO algorithms while being computationally faster.

1 Introduction

The context of this work is continuous black-box Multi-Modal Optimization (MMO). Given an objective function F, the goal of MMO is to discover **all** global optima of F (and not just a few of them). Because of the "black-box" hypothesis, this paper focuses on gradient-free methods, more particularly Evolution Strategies (ES) [5], addressing both theoretical and experimental issues.

Note that there are to-date very few rigorous analyses of ES for MMO. But it has been experimentally shown that in the MMO setting, choosing a large population size λ reduces the risk being trapped in local minima [27, 9, 1]. Unfortunately, it has also been shown that some classical ES are not very efficient for large λ [4], and that self-adaptive ES are quite fast and reliable in that case, with a good speed-up as a function of λ [4].

Several MMO-specific methods have been proposed in the evolutionary framework, and their precise description cannot be included here for space considerations. Please refer to the survey proposed in [23] (or to the original papers of course). A popular family of MMO algorithms is based on niching techniques: sharing [8]; clearing [22], including the modified clearing approach proposed in [23]; crowding [6], including deterministic [18] and probabilistic [19] versions; clustering [30]; species conserving genetic algorithms [16]; and finally, different

restart strategies, to the recent state-of-the-art restart with increasing population size [1]. Several other works have been devoted to MMO outside the evolutionary community. For instance, EGO [12], IAGO [29] provide very efficient algorithms in terms of precision/number of fitness evaluations, but are computationally far too expensive unless the objective function is itself very costly (requiring hours or days of computation per point). Moreover, they have no theoretical guarantee in spite of their elegant derivation. UNLEO [3] proposed an approximation of optimal optimization algorithm under robustness constraints, with nice theoretical guarantees; however, it is, too, far too expensive, and could hence be tested with a few tenths of function evaluations.

This paper proposes a very simple restart strategy that can encapsulate any (local) optimization algorithm, and for which convergence rates can be theoretically derived. This strategy is based on quasi-random restarts, and a decreasing schedule for some 'step-size' parameter. Comparative experiments with the extensive results published in [23] are used to demonstrate the applicability and efficiency of the proposed strategy, in the case where the embedded local algorithm is a simple (1+1)-ES with 1/5th adaptation rule. However, the proposed restart strategy heavily relies on a specific parameter d (see Alg. 1), somehow analogous to a parameter used in the modified clearing which outperformed by far all other algorithms [23]. Hence, because this parameter d will be tuned for the problems used in the experiments, no fair comparison can be done with other published results for which no such parameter tuning was performed. Our conclusions will therefore be limited to the design of a generic restart algorithm based on quasi-random sequences with theoretical guarantees and tight complexity bounds (see Corollary 1), that outperforms uniform restarts, and reaches state of the art performance provided a good setting of parameter d, which essentially quantifies some prior knowledge about the minimal distance between two optima.

The remaining of the paper is organized in the following way: Section 2 introduces the notations used throughout the paper, and surveys the state-of-the-art in quasi-random (QR) sequences. Section 3 introduces the proposed restart strategy, and rigorously quantifies the improvement provided by QR points in the initialization of MMO and the requirement that the initial step-size after a restart goes to zero as the number of restarts increases. Section 4 provides comparative experimental results with state-of-the-art MMO methods as well as the simple random restart method with constant step-size.

2 Mathematical Background

Notations: For each E, subset of a topological space, $\overline{E}$ will denote the *topological closure* of E, i.e., the intersection of all closed sets containing E, and $\#\mathbb{E}$ denote the number of elements of E. For each sequence $S = s_1, s_2, s_3, \dots$ of points in a topological space, the *accumulation* of S is defined as $Acc\ S = \cap_{n\geq 1}\overline{\{s_{n+1}, s_{n+2}, \dots\}}$. For each sequence $X = x_1, x_2, \dots$ of points in $[0,1]^D$, the *dispersion* of X is defined as $Disp(X,n) = \sup_{x\in[0,1]^D} \inf_{i\in[[1,n]]} ||x - x_i||$.

Quasi-random (QR) points A quasi-random sequence is a (possibly randomized) sequence with some uniformity properties that make it, intuitively, "more uniform" than a pseudo-random sequence. After astonishing results in numerical integration (convergence in $1/n$ instead of $1/\sqrt{n}$ for numerical integration, within logarithmic factors) and successful experiments in random searchs [20], QR points have been used in several works dealing with evolution strategies, during initialization [13, 7] or mutation [2, 26, 25]. Furthermore, "modern" QR sequences using scrambling [14] have been demonstrated to outperform older QR sequences [26, 25]. Hence, following [28, 25], this paper will only consider Halton sequences with random scrambling, and this Section will only briefly introduce them – please refer to [20, 21] for more details and references.

Let us first define Van Der Corput's sequence: Given a prime number p, the n^{th} element $vdc_{n,p}$ of the Van Der Corput sequence in basis p is defined by:

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

A classical improvement in terms of discrepancy for moderate values of n and large values of d, termed *scrambling*, defines $vdc_{n,p}$ as $vdc_{n,p} = 0.\pi(d_1)\pi(d_2)\dots\pi(d_k)$ where π is some permutation of $[[0, p-1]]$ such that $\pi(0) = 0$ in order to ensure $\forall n, vdc_{n,p} \neq 0$.

Halton sequences generalize Van Der Corput sequences to dimension D by using one different prime number per dimension. Let $p_i, i \in [[1, D]]$ be D prime numbers. The n^{th} element h_n of a Halton sequence in dimension D is defined by $h_n = (vdc_{n,p_1}, vdc_{n,p_2}, \dots, vdc_{n,p_D}) \in [0, 1]^D$. Scrambled-Halton sequences, like the ones used in this paper, are scrambled using a randomly drawn permutation for each $i \in [[1, D]]$.

The N^{th} Hammersley point set is $\{\text{HAMM}_{N,1}, \text{HAMM}_{N,2}, \dots, \text{HAMM}_{N,N}\}$, where $\text{HAMM}_{N,n} = ((n-1)/N, vdc_{n,p_1}, vdc_{n,p_2}, \dots, vdc_{n,p_{D-1}})$.

3 Theoretical Analysis of a Simple Restart Strategy

Let $\mathcal{D}$ be the optimization domain, embedded in a normed vector space (with norm $||.||$). Let $\mathcal{F}$ be a family of fitness functions, i.e., of mappings from $\mathcal{D}$ to $\mathbb{R}$. For any fitness function $f \in \mathcal{F}$, let $X^*(f)$ be the set of interest[1]. Let ES be an optimization algorithm that takes as input $x \in \mathcal{D}$ and $\sigma > 0$ (initial point and step-size respectively), depends on f, and outputs some $x' = ES(x, \sigma, f) \in \mathcal{D}$ [2]. Think of σ as a radius at which local optima are searched; however, it is not necessary to assume that ES always finds an optimum x' at distance $< \sigma$ of x, but only that for σ sufficiently small and x sufficiently close to x^*, $x' = ES(x, \sigma, f) = x^*$. Finally, for any given sequence $S = (x_i, \sigma_i)_{i \in \mathbb{N}} \in (\mathcal{D} \times]0, \infty[)^{\mathbb{N}}$, denote $RS(S)$ the restart algorithm that successively starts from (x_1, σ_1), (x_2, σ_2), ...

[1] in most cases, $X^*(f)$ will be the set of local optima of f, though the results below have some generality w.r.t. $X^*(f)$.

[2] x' is the output of a whole run of ES: in general, it will be the best point of the run; however, here again the results are more general.

Definitions:
(i) ES has the **convergence property** if

$$(\forall f \in \mathcal{F})\ (\forall (x,\sigma) \in \mathcal{D}\times]0,\infty[)\ (ES(x,\sigma,f) \in X^*(f)) \tag{1}$$

(ii) ES has the **locality property** w.r.t. $\mathcal{F}$ if

$$\begin{array}{l}(\forall x^* \in X^*(f))\ (\exists \epsilon > 0)\ (\exists \sigma_0 > 0) \text{ s.t.} \\ (||x - x^*|| < \epsilon) \text{ and } (0 < \sigma < \sigma_0) \Rightarrow (ES(x,\sigma,f) = x^*).\end{array} \tag{2}$$

Note that this does not imply that $ES(x,\sigma,f^*)$ is necessarily within distance σ of x, which would be an unrealistic assumption.

(iii) ES has the **strong locality property** w.r.t. $\mathcal{F}$ if

$$\begin{array}{l}(\exists \epsilon > 0)\ (\exists \sigma_0 > 0) \text{ s.t. } (\forall f \in \mathcal{F})\ (\forall x^* \in X^*(f)) \\ (||x - x^*|| < \epsilon) \text{ and } (0 < \sigma < \sigma_0) \Rightarrow ES(x,\sigma,f) = x^*.\end{array} \tag{3}$$

(iv) For any sequence $S \in (\mathcal{D}\times]0,\infty[)^{\mathbb{N}}$, the restart algorithm $RS(S)$ is said to be **consistent w.r.t.** $\mathcal{F}$ if

$$(\forall f \in \mathcal{F})\ \{ES(x_i,\sigma_i,f); i \in \mathbb{N}\} = X^*(f).$$

We can now state the following consistency theorem.

Theorem 1 (Consistency of the restart algorithm). *Assume that ES has the convergence property (Eq. 1) and the locality property (Eq. 2). Then:*

1. *If $(X^*(f) \times \{0\} \subset Acc\ S)$ for all $f \in \mathcal{F}$, then $RS(S)$ is consistent w.r.t. $\mathcal{F}$;*
2. *If $\mathcal{D} \times \{0\} \subset Acc\ S$, then $RS(S)$ is consistent w.r.t. all $\mathcal{F} \subset \mathbb{R}^{\mathcal{D}}$.*

Proof: (2) is an immediate consequence of (1) so let us prove (1).
Let $f \in \mathcal{F}$. Eq. 1 immediately implies that $\{ES(x_i,\sigma_i,f); i \geq 1\} \subset X^*(f)$.
Let $x^* \in X^*(f)$; then $(x^*,0)$ is in $Acc\ S$ by assumption. Using Eq. 2, it follows that $x^* = ES(x_i,\sigma_i,f)$ for some $i \geq 1$; this proves that $X^*(f) \subset \{ES(x_i,\sigma_i,f); i \geq 1\}$; hence the expected result. □

Remark 1 The assumption $X^*(f) \times \{0\} \subset Acc\ S$ is necessary. It holds in particular with random restarts points and random initial step-sizes with non-zero density close to 0; or random restart points and step-sizes decreasing to 0; in both cases, quasi-random restarts can be used instead of random restarts.

Next result now considers the number of restarts required for finding all points in $X^*(f)$. Sequences of starting points are now considered stochastic, hence the expectation operator:

Proposition 1 (QR restarts are faster). *Assume that ES has the convergence property (Eq. 1) and the strong locality property (Eq. 3) for some ϵ, σ_0. Assume that $\sigma_i = \sigma_0$ for all $i \geq 1$. Define*
$\#RS(\mathcal{F}) = \sup_{f\in\mathcal{F}} \mathbb{E}[\inf\{n \in \mathbb{N}; X^*(f) = \{ES(x_1,\sigma_1,f),\ldots,ES(x_n,\sigma_n,f)\}\}]$.
Then, $\#RS(\mathcal{F}) \leq \mathbb{E}[\inf\{n \in \mathbb{N}; Disp((x_i)_{i\in\mathbb{N}},n) \leq \epsilon\}]$.

Proof: By Eq. 1, $X^*(f) \subset \{ES(x_1, \sigma_1, f), \dots, ES(x_n, \sigma_n, f)\}$ for all $n \in \mathbb{N}$. On the other hand, if $Disp((x_i)_{i\in\mathbb{N}}, n) < \epsilon$, then Eq. 3 implies

$$\#RS(\mathcal{F}) \leq \sup_{f\in\mathcal{F}} \mathbb{E}[\inf\{n \in \mathbb{N}; X^*(f) \subset B(x_1, \epsilon) \cup B(x_2, \epsilon) \cup \dots \cup B(x_n, \epsilon)\}]$$

and this is at most n; hence the expected result. □

The bound is tight for some simple cases, e.g., optima distributed on a grid with sphere-like functions on Voronoi-cells built on the optima, and ES converging locally:

$$X^*(f) = \{(k_1\epsilon, k_2\epsilon, \dots, k_D\epsilon); (k_1, \dots, k_D) \in [[0, \lfloor 1\epsilon \rfloor]]^D\}. \tag{4}$$

The dispersion $Disp$ can therefore be used for quantifying the quality of a sequence of restart points. The optimal dispersion is reached by some grid-based sampling (e.g. Sukharev grids [17]) reaching $Disp(x, n) = O(n^{1/D})$. The **dispersion complexity**, i.e., the number n of points required for reaching dispersion ϵ is then[3]:

$O\left((1/\epsilon)^D \log(1/\epsilon)\right)$ for all low-discrepancy sequences; th. 6.6 p152, (C1)
$O\left((1/\epsilon)^D\right)$ for Halton or Hammersley (Section 2); th. 6.12+6.13 p157, (C2)
$\Omega\left((1/\epsilon)^D\right)$ for all sequences or point sets; th. 6.8 p154, (C3)
$\Omega\left((1/\epsilon)^D \log(1/\epsilon)\right)$ for random sequences (expected value; see Theorem 2) (C4)

The results above are based on the notion of discrepancy. In particular, low-discrepancy (also termed quasi-random) sequences have *extreme discrepancy* $O(\log(n)^D/n)$. However, only complexity result (C1) will be necessary here.

To the best of our knowledge, complexity (C4) for random sequences has not yet been published. It can nonetheless easily be derived by reduction to the classical Coupon collector theorem (proof omitted for space reasons):

Theorem 2 (Dispersion of random points). *Consider $x_1, \dots, x_n, \dots$ randomly independently uniformly distributed in $[0, 1]^D$ and $\epsilon > 0$. Then $\mathbb{E}[inf\{n \in \mathbb{N}; Disp(x, n) < \epsilon] = \theta\left(1/\epsilon^D \log(1/\epsilon)\right)$.*

From Proposition 1, and the complexity of dispersions above, it comes

Corollary 1 (Complexity in terms of number of restarts). *Let $(\mathcal{F}^{(k)})_{k\in\mathbb{N}}$ be a family of sets of fitness functions defined on $[0, 1]^D$ for some D. Suppose that for each $k \in \mathbb{N}$, $\mathcal{F}^{(k)}$ has strong local property (Eq. 3) with values $\epsilon^{(k)}, \sigma_0^{(k)}$. Then, for some $C > 0$, a quasi-random restart with Halton sequence[4] and $\sigma_i < \sigma_0$ ensures that*

$$(\forall k \in \mathbb{N}), (\forall f \in \mathcal{F}^{(k)}), (X^*(f) \subset \{ES(x_i, \sigma_i, f); i \in [[1, C/\epsilon^D]]\}). \tag{5}$$

This is not true for random restart, and C/ϵ^D cannot be replaced by $o(1/\epsilon^D)$.

[3] all references are to be found in [20]. See references therein for more details.
[4] or any sequence with dispersion complexity (C1)

Algorithm 1 This generic algorithm includes Random restart with Decreasing Step-size (RDS) as well as Quasi-Random restart with Decreasing Step-size (QRDS) algorithm. Constant, linearly or quadratically decreasing versions can be implemented on line 5. The case with murder is the case $d > 0$ (line 14).

Require: $(x_1, x_2, \ldots)$ sequence of starting points, (d, σ^*) precision thresholds
1: $n = 0, optimaFound = \emptyset$
2: **while** Maximum number of evaluations not reached, and all optima not found **do**
3: $n \leftarrow n + 1$
4: $y = x_n$ // nth point the sequence of starting points
5: $\sigma = nextSigma(n, \sigma_0)$ // constant or decreasing σ
6: $State \leftarrow alive$
7: **while** $State = alive$ **do**
8: $y' = y + \sigma\mathcal{N}(0, I_d)$ // $\mathcal{N}(0, I_d)$ is an isotropic Gaussian random variable
9: **if** y' better than y **then**
10: $y = y'$; $\sigma = 2\sigma$
11: **else**
12: $y = y'$; $\sigma = 2^{-1/4}\sigma$
13: **for all** $(opt \in optimaFound)$ **do**
14: **if** $(||y - opt|| < d)$ or $(\sigma < \sigma^*)$ **then**
15: State$\leftarrow$dead
16: $optimaFound = optimaFound \cup \{y\}$

Proof: Eq. 5 is a consequence of (C1) and Proposition 1.

The fact that this is not true for random restart is the application of complexity (C4) to the particular case shown above (Eq. 4).

Finally, $\Omega(1/\epsilon^D)$ restarts are needed in order to find the $\Omega(1/\epsilon^D)$ optima in Eq. 4: it is hence not possible in the general case to replace C/ϵ^D by $o(1/\epsilon^D)$. □

4 Experimental results

This section presents comparative experimental results for some particular instances of restart algorithms to which the theoretical results of above Section 3 can be applied. The main ingredient of a restart algorithm is its embedded 'local' optimizer. The choice made here is that of an Evolution Strategy, for the robustness of this class of algorithm. However, only local convergence properties are required, far from any sophisticated variant designed for multimodal fitness functions for instance [27, 9, 1]. Furthermore, the testbed we want to compare to [23] does not require large values of λ, nor covariance matrix adaptation. Hence the simple (1+1)-ES with $\frac{1}{5}^{th}$-rule was chosen, as it gives very good results in very short time according to some preliminary runs. Finally, a (1+1)-ES satisfies the hypotheses of Theorem 1, and hence all results of Section 3 apply depending only on the properties of the sequence of starting points and initial step-sizes.

The precise instances of restart algorithm under scrutiny here are defined in Algorithm 1: this includes random and quasi-random sequences of starting points in the domain given as input (line 4), as well as different strategies for the step-size change from one run to the other, depending on line 5: function *nextSigma* can return a constant value σ_0, a linearly decreasing value $(\sigma_0/(n+1))$ or a quadratically decreasing value $(\sigma_0/(n+1)^2)$.

One run of the local algorithm can be killed when either the step-size goes beyond a given thresholds σ^* (defaulted to 10^{-6} unless otherwise stated), or

when the current solution gets too close to a previously discovered local optimum, up to a tolerance d (line 14). In the latter case, the algorithm is said *with murder*. But the choice of the threshold distance d is not trivial: a too small d wastes time, and too large values give imprecise results. Parameter d is a clear and strong drawback of the proposed algorithms, as the results are very sensitive to the value of d. Note however that the same remark holds for the modified clearing approach which outperformed by far all other methods in [23]. Nevertheless, because of this parameter, the generality of the results shown below is limited. In particular, we will limit our claims to the comparison with the results in [23], obtaining results that are comparable with the very good results of the modified clearing, with a similar parameter d, but with a simple restart algorithm.

Let us define K as the number of optima for the murder operator (the murder operator is not required for those complexity bounds). Let us first compare the complexity of the proposed restart algorithm with those provided in [23] (Table 1 (left)). The complexity for RDS/QRDS is immediate. In the case of murder operator, the complexity bound holds provided that K and d are such that the proportion of the domain which is forbidden by the murder operator is never larger than a fixed proportion of the whole domain. λ is the population size, 1 in our case (but the complexity results hold for any value of λ). RTS [10, 11] and SCGA [15] stand for Restricted Tournament Selection and Species Conserving Genetic Algorithm respectively. w is window size of RTS. N_c is the number of clusters.

Algorithm	Complexity
RDS/QRDS, Clearing	$\Theta(\lambda)$
Deterministic or probabilistic crowding	$\Theta(\lambda)$
RTS	λw
QRDS/RDS with murder	$O(\lambda K)$
SCGA	$\Omega(\lambda)$ and $O(\lambda^2)$
Sharing, modified clearing	$\Theta(\lambda^2)$
Clustering	$\Theta(\lambda N_c)$

	Number of evaluations	Number of restarts	Computational overhead for RDS vs QRDS
Constant initial step-size $\sigma_0 = 0.1$			
RDS	1652	13.7	1%
QRDS	1628	13.41	
Constant initial step-size $\sigma_0 = 0.01$			
RDS	740 ± 15.81	6.07 ± 0.23	64%
QRDS	452 ± 8.95	2.3 ± 0.11	
Constant initial step-size $\sigma_0 = 0.001$			
RDS	808 ± 18.37	6.44 ± 0.25	79%
QRDS	451 ± 8.54	2.21 ± 0.10	
Quadratically decreasing step-size, $\sigma_0 = 1.$			
RDS	726 ± 18.34	6.73 ± 0.28	46%
QRDS	498 ± 10.15	3.72 ± 0.15	
Quadratically decreasing step-size, $\sigma_0 = 0.1$			
RDS	755 ± 17.40	6.65 ± 0.25	64%
QRDS	461 ± 8.74	2.79 ± 0.11	

Table 1. Left: Complexity of various algorithms. Most results are from [23] (see text). Right: Results on the sine function in dimension 1, for RDS and WRDS, and different strategies for σ. The last column is the percentage of additional evaluations when using random instead of quasi-random.

Let us then test the different approaches on the test functions provided in [23]. The first function is a n-dimensional sine, with 5^n peaks, defined on $[0, 1]$ as$f(x) = 1 - \frac{1}{n}\sum_{i=1}^{n}(1 - \sin^6(5\pi x_i))$. In all experiments with this function, a

point x^* will be considered an optimum if $f(x^*) > 0.997$.
Another important test function is the hump function defined in [23] as: $f(x) = h \max(1 - (\inf_k ||x - x_k||/r)^{\alpha_k}, 0)$where $\alpha_k = 1$, $r = 1.45$, $h = 1$; sequence $x_1, \ldots, x_k$ is randomly drawn in domain $[0, 1]^D$; the problem is used in dimension 25 with $k = 50$, the most difficult case according to [23].

4.1 Constant initial step-size is dangerous

First introduced in [24], the idea of decreasing σ at each restart is somewhat natural, when considering the convergence proof (see Remark 1). From the results on the sine function in dimension 1 (Table 1-right), it is clear that a too large fixed σ leads to poor results, while a too small σ also hinders the performances. Indeed, when looking for all optima, a large initial step-size might be helpful in order to avoid poor local optima. However it is a bad idea for finding optima close to the frontier of the domain when there are big basins of attractions. Furthermore, the proved version, with quasi-random restarts and decreasing σ, performs well without any tuning, though less efficiently than the version with *a posteriori* chosen fixed σ. Yet, its good performances independently of any parameter tuning is a strong argument for the proved method.

4.2 Validating quasi-random sequences

Let us now focus on the comparison between the use of quasi-random vs random restarts, i.e., RDS vs QRDS. From results on the multi-dimensional sine function (Tables 1-right and 2), it is clear that quasi-random becomes more and more efficient as the number of optima increases.

Dimension D	RDS	QRDS	Additional cost for RDS over QRDS
3	143986 (803)	109128 (609)	32%
2	11673 (98)	8512 (71)	37%
1	777 (13)	447 (8)	74%

Table 2. RDS vs QRDS on the multidimensional sine function: number of evaluations (number of restarts), averaged over 30 runs, for finding the 5^D optima in dimension D. Here the step-size is quadratically decreasing with initial step-size $\sigma_0 = 0.1$.

Let us now revisit the sine function in dimension 1, and increase the number of optima by increasing its parameter K from 5 to 50, 500, and 1000. Other parameters of the algorithms are here $\sigma_{init} = 10^{-1}/K$, $\sigma^* = 5.10^{-4}/K$, and $d = 0.5/K$ for the murder threshold. The performances reported in Table 3-left witness the computational effort until all optima are found, i.e. there is a point of fitness > 0.997 at distance lower than half the distance between optima (please remember than 0.997 is the chosen threshold in this benchmark). Those results show that QRDS becomes more and more efficient when compared to RDS as the number of optima increases. Furthermore, the murder operator is clearly highly efficient.

	Nb of evaluations	Nb of restarts	Ratio
5 optima			
RDS	1484 ± 33.59	5.79 ± 0.24	25 %/
QRDS	1187 ± 30.27	4.17 ± 0.20	152%
QRDS+M	588 ± 3.24	1.31 ± 0.04	
50 optima			
RDS	30588 ± 156.28	171.45 ± 1.12	46% /
QRDS	20939 ± 100.04	102.24 ± 0.71	364%
QRDS+M	6583 ± 2.39	17.39 ± 0.05	
500 optima			
RDS	470080 ± 548.60	2900.75± 3.96	67% /
QRDS	281877± 279.15	1540.5± 2.01	603%
QRDS+M	66789 ± 3.07	161.92 ± 0.07	
1000 optima			
RDS	1009144 ± 747.05	6298.16 ± 5.40	61 % /
QRDS	627696 ± 409.41	3545.61 ± 2.96	655%
QRDS+M	133587 ± 3.11	320.8 ± 0.07	

Algorithm	Nb of optima found
Sharing	$\simeq 0$
D./P. Crowding	$\simeq 0$ / 0
RTS	$\simeq 0$
SCGA	$\simeq 0$
Clustering	$\simeq 0$
Clearing	$\simeq 43$
Modified Clearing	$\simeq 50$
RDS	49.92 (over 100 runs)
QRDS	49.95 (over 100 runs)

Table 3. Left: Performance of RDS, QRDS, and QRDS with Murder when the number of optima on the 1-D sine function increases. The last column shows the percentage of additional evaluations for RDS over the given algorithm. Right: Comparative results on the 25-dimensional hump problem. All results but RDS/QRDS are taken from [23], where the modified clearing is reported to have found all 50 optima in all of the 30 runs. RDS and QRDS being much faster (see Table 1), 100 experiments have been run easily, and almost all 50 optima have been consistently found. See text for details.

4.3 Comparison with Modified Clearing

In this section, we compare the restart as previously defined (quasi-random restarts, decreasing σ and murder) to the best techniques in the survey [23]. Interestingly, some algorithms tested in [23] (Probabilistic Crowding and SCGA) could not even find all optima of the simple sine function in dimension 1. Note that QRDS finds all optima within a few hundred evaluations, whereas according to [23] nearly 100 generations of population size 50 are necessary for finding the 5 optima for all methods. However, because [23] does not provide quantitative results, precise comparisons are not possible here. Therefore, the following comparative results use the hump function, for which [23] provides extensive experiments with detailed results. This function is particularly challenging, because [23] points out that no algorithm except their modified clearing can solve the problem. The modified clearing, however, finds all optima in each of the 30 runs, whereas all methods (as reported in Table 1), except clearing, do not even find a single optimum. Table 3-right reports the results of RDS and QRDS for the hardest instance, in dimension 25 where the number of optima is 50. Averaged over 100 runs, both methods find almost all 50 optima (on average more than 49.9).

5 Conclusion

This paper has introduced some generic framework for restart algorithms embedding a (local) optimization algorithm. With limited hypotheses about the

local properties of the embedded algorithm, in particular with respect to some initial parameter σ describing the size of its basins of attraction, we have proved some convergence properties with speed depending on the dispersion of the sequence of starting points, independently of any other parameter of the embedded algorithm.

Actual instances of the generic algorithm have then been proposed, embedding a (1+1)-Evolution Strategy with $\frac{1}{5}^{th}$ rule, for which the initial step-size plays the role of parameter σ above. Random and quasi-random sequences of starting points can be used. The proposed algorithms have mathematically proved convergence properties. Thanks to the decreasing of the initial step-size to 0, the convergence is proved independently of initialization parameters. Furthermore, experimental validation of the proposed algorithm have been conducted, comparing random and quasi-random sequences of starting points, and judging performance with respect to other previously published algorithms [23].

A first conclusion is the not surprising superiority of quasi-random restarts over random restarts. The improvement due to QR points is moderate (a logarithmic factor), but regular and increasing when considering more complicated cases (more optima to be found). We have no experimental evidence in this paper or in the literature for the superiority of any algorithm (whatever complicated and computationally expensive) over the simple restart algorithm with decreasing σ and quasi-random initializations that has been proposed here. QRDS performed equally or better than modified clearing [23], whilst keeping a linear computational cost. All methods with non quadratic cost benchmarked in [23] are much less efficient than QRDS, RDS or modified clearing, while RDS and QRDS are much cheaper than modified clearing.

The murder operator, that kills runs that get close to previously found optima, was found highly beneficial (up to more than 700 % speed-up for 500 optima). However, QRDS with murder operator has the same weakness than modified clearing: it introduces a crucial parameter taking into account the distance between different optima, somewhat similar to σ_{clear} parameter of modified clearing (that has additionally two arbitrary constants 1.5 and 3 for solving the difficult hump function).

Let us now discuss the limitations of our analysis, and some general elements. An important element in this work is the choice of benchmark functions. The hump function is in fact a distribution of fitness functions, and not only a fitness function, or random translations of a fitness function. We agree with the authors of [23] that solving the hump function is by no means an easy task. However, this benchmark has its limitations: as the function is locally very simple, and as a very loose precision is required, the best choice for the embedded ES is the 1+1-ES with a very loose halting condition: this is perhaps not the less realistic scenario, but this certainly does not covers all cases. A strong advantage of the hump function is that it cannot easily be overfitted. However, all successful methods on the hump function have a parameter which is intuitively related to the distance between optima: This suggests that, in spite of the random part in the hump function, some overfitting nevertheless happens when tuning an

algorithm on the hump function. Also, we feel these experiments are definitely convincing regarding the importance of mathematical analysis in MMO: it is otherwise very easy to conclude positively about an algorithm, without seeing its precise limitations. Thanks to theoretical analysis, we could clearly see that decreasing the step-size provides a real advantage (a decreasing step-size provides consistency) and that quasi-randomizing the restarts provides an improvement - and to the best of our knowledge, as QRDS has the best experimental results and the best proved results, this paper provides a clear and simple baseline for future research in MMO. A case in which clearing approaches might perform better than the proposed approach is the parallel case. The murder operator might be less efficient if several populations evolve simultaneously. The proposed approach has been designed for the case of MMO in which the goal is to locate all optima, opposed to the case where the goal is to locate the global optimum, but there exist many local optima. However, these two forms of MMO are probably not so different - if the local optima have fitness value close to the one of the global optimum x^*, we have to check all local optima.

References

1. A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*, pages 1769–1776, 2005.
2. A. Auger, M. Jebalia, and O. Teytaud. XSE: quasi-random mutations for evolution strategies. In *Proceedings of EA'05, 12 pages*, 2005.
3. A. Auger and O. Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.
4. H. Beyer and B. Sendhoff. Covariance Matrix Adaptation Revisited–The CMSA Evolution Strategy–. *Parallel Problem Solving from Nature–PPSN X*, pages 123–132, 2008.
5. H.-G. Beyer. *The Theory of Evolution Strategies*. Springer, Heidelberg, 2001.
6. K. DeJong. *The Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Harbor, 1975. *Dissertation Abstract International*, 36(10), 5140B. (University Microfilms No 76-9381).
7. A. Georgieva and I. Jordanov. A hybrid meta-heuristic for global optimisation using low-discrepancy sequences of points. *Computers and Operations Research, - special issue on hybrid metaheuristics*, In press.
8. D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodalfunction optimization. In J. J. Grefenstette, editor, *ICGA*, pages 41–49. Lawrence Erlbaum Associates, 1987.
9. N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer, 2004.
10. G. Harik. Finding multiple solutions in problems of bounded difficulty. Technical Report 94002, University of Illinois at Urbana Champaign, 1994.
11. G. Harik. Finding multimodal solutions using restricted tournament selection. In L. J. Eshelman, editor, *Proc. Sixth International Conference on Genetic Algorithms*, pages 24–31. Morgan Kaufmann, 1995.

12. D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 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*, pages 419 – 474. Kluwer Academic, 2002.
15. J. Li, M. Balazs, G. Parks, and P. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary computation*, 10(3):207–234, 2002.
16. J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.*, 10(3):207–234, 2002.
17. S. R. Lindemann and S. M. LaValle. Incremental low-discrepancy lattice methods for motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2920–2927, 2003.
18. S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1995.
19. O. J. Mengshoel and D. E. Goldberg. Probabilistic crowding: Deterministic crowding with probabilisitic replacement. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 409–416, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.
20. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia: SIAM, 1992.
21. A. B. Owen. Quasi-Monte Carlo sampling. In H. W. Jensen, editor, *Monte Carlo Ray Tracing: Siggraph 2003 Course 44*, pages 69–88. SIGGRAPH, 2003.
22. A. Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *International Conference on Evolutionary Computation*, pages 798–803, 1996.
23. G. Singh and D. Kalyanmoy Deb. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1305–1312, New York, NY, USA, 2006. ACM.
24. F. Teytaud and O. Teytaud. Log(lambda) Modifications for Optimal Parallelism. In *Parallel Problem Solving From Nature–PPSN XI*, Krakow Pologne, 09 2010.
25. O. Teytaud. When does quasi-random work? *Parallel Problem Solving from Nature–PPSN X*, pages 325–336, 2008.
26. O. Teytaud and S. Gelly. DCMA: yet another derandomization in covariance-matrix-adaptation. In D. Thierens et al., editor, *ACM-GECCO'07*, pages 955–963, New York, NY, USA, 2007. ACM.
27. F. van den Bergh and A. Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers, 2000.
28. 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.
29. J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534, 2009.
30. X. Yin and N. Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In R. F. Albrecht, N. C. Steele, and C. R. Reeves, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 450–457, Wien, 1993. Springer Verlag.