Investigating the Local-Meta-Model CMA-ES for Large Population Sizes

Zyed Bouzarkouna, Anne Auger, Didier Yu Ding

Investigating the Local-Meta-Model CMA-ES for Large Population Sizes

Zyed Bouzarkouna[1,2], Anne Auger[2], and Didier Yu Ding[1]

[1] IFP (Institut Français du Pétrole)
1,4 avenue de bois préau
92852 Rueil-Malmaison Cedex, France
[2] TAO Team INRIA Saclay-Ile-de-France
LRI Paris Sud University
91405 Orsay Cedex, France

Abstract. For many real-life engineering optimization problems, the cost of one objective function evaluation can take several minutes or hours. In this context, a popular approach to reduce the number of function evaluations consists in building a (meta-)model of the function to be optimized using the points explored during the optimization process and replacing some (true) function evaluations by the function values given by the meta-model. In this paper, the local-meta-model CMA-ES (lmm-CMA) proposed by Kern et al. in 2006 coupling local quadratic meta-models with the Covariance Matrix Adaptation Evolution Strategy is investigated. The scaling of the algorithm with respect to the population size is analyzed and limitations of the approach for population sizes larger than the default one are shown. A new variant for deciding when the meta-model is accepted is proposed. The choice of the recombination type is also investigated to conclude that the weighted recombination is the most appropriate. Finally, this paper illustrates the influence of the different initial parameters on the convergence of the algorithm, for multimodal functions.

Key words: Optimization, Covariance Matrix Adaptation - Evolution Strategy (CMA-ES), Meta-models

1 Introduction

Many real-world optimization problems are formulated in a black-box scenario where the objective function to optimize $f : \mathbb{R}^n \mapsto \mathbb{R}$ may have noise, multiple optima and can be computationally expensive. Evolutionary algorithms (EAs) are stochastic population based optimization algorithms that are usually a good choice to cope with noise and multiple optima. For expensive objective functions–several minutes to several hours for one evaluation–a strategy is to couple EAs with meta-models: a model of f is built, based on "true" evaluations of f, and used during the optimization process to save evaluations of the expensive objective function [1]. One key issue when coupling EAs and meta-models is to decide when the quality of the model is good enough to continue exploiting

this model and when new evaluations on the "true" objective functions should be performed. Indeed, performing too few evaluations on the original objective function can result in suboptimal solutions whereas performing too many of them can lead to a non efficient approach.

The covariance matrix adaptation ES (CMA-ES) [2, 3] is an EA recognized as one of the most powerful derivative-free optimizers for continuous optimization[3]. At each iteration of CMA-ES, the evaluation of candidate solutions on the objective function are performed. However, from those evaluations only the ranking information is used. In consequence the algorithm is invariant to transformations on f preserving the ranking. CMA-ES was coupled with local meta-models by Kern et al. [4]. In the proposed algorithm, lmm-CMA, the quality of the meta-model is appraised by tracking the change in the *exact ranking* of the best individuals. The lmm-CMA algorithm has been evaluated on test functions using the default population size of CMA-ES for unimodal functions and for some multimodal functions and has been shown to improve CMA-ES [4].

In this paper, we analyze the performance of lmm-CMA when using population sizes larger than the default one. We show that tracking the exact rank-change of the best solutions to determine when to re-evaluate new solutions is a too conservative criterion and leads to a decrease of the speedup with respect to CMA-ES with increasing population size. Instead we propose a less conservative criterion that we evaluate on test functions. This paper is structured as follows. Section 2 gives an overview of CMA-ES and lmm-CMA. In Section 3, we evaluate lmm-CMA-ES for population size larger than the default one. In Sections 4.1 and 4.2, we propose a new variant of lmm-CMA. In Section 4.3, the influence of the recombination type on the new variant is tested. The influence of initial parameters is analyzed in Section 4.4.

2 CMA-ES with local meta-models

The Covariance Matrix Adaptation ES CMA-ES is a stochastic optimization algorithm where at each iteration g, a population of λ points is sampled according to a multivariate normal distribution. The objective function of the λ points is then evaluated and the parameters of the multivariate normal distribution are updated using the feedback obtained on the objective function. More specifically, let $(\mathbf{m}^g, g \in \mathbb{N})$ be the sequence of mean values of the multivariate normal distribution generated by CMA-ES, constituting the sequence of estimate of the optimum and let $(\sigma^g, g \in \mathbb{N})$ and $(\mathbf{C}^g, g \in \mathbb{N})$ be respectively the sequences of step-sizes and covariance matrices. Assume that $\mathbf{m}^g, \sigma^g, \mathbf{C}^g$ are given, new points or *individuals* are sampled according to:

$$\mathbf{x}_i^g = \mathbf{m}^g + \sigma^g \mathcal{N}_i(0, \mathbf{C}^g), \qquad \text{for } i = 1 \dots \lambda \ , \tag{1}$$

where $(\mathcal{N}_i(0, \mathbf{C}^g))_{1 \leq i \leq \lambda}$ are λ independent multivariate normal distributions with zero mean vector and covariance matrix $\mathbf{C}^g$. Those λ individuals are ranked

[3] See http://coco.gforge.inria.fr/doku.php?id=bbob-2009-results.

according to f:

$$f(\mathbf{x}^g_{1:\lambda}) \leq \ldots f(\mathbf{x}^g_{\mu:\lambda}) \leq \ldots f(\mathbf{x}^g_{\lambda:\lambda}) \tag{2}$$

where we use the notation $\mathbf{x}^g_{i:\lambda}$ for i^{th} best individual. The mean $\mathbf{m}^g$ is then updated by taking the weighted mean of the best μ individuals, $\mathbf{m}^{g+1} = \sum_{i=1}^{\mu} \omega_i \mathbf{x}^g_{i:\lambda}$, where in general $\mu = \frac{\lambda}{2}$ and $(w_i)_{1\leq i\leq \mu}$ denote strictly positive and normalized weights, i.e., satisfying $\sum_{i=1}^{\mu} \omega_i = 1$. The default weights are equal to:

$$\omega_i = \frac{\ln(\mu+1) - \ln(i)}{\mu \ln(\mu+1) - \ln(\mu!)}, \qquad \text{for } i = 1 \ldots \mu. \tag{3}$$

Furthermore σ^g and $\mathbf{C}^g$ are updated as well after evaluation and we refer to [3] for the equation updates. All updates rely on the ranking determined by Eq. 2 only and not on the exact value of the objective functions such that the CMA-ES is invariant when optimizing f or $g \circ f$ where $g : \mathbb{R} \mapsto \mathbb{R}$ is a strictly increasing mapping. The default population size λ equals $4 + \lfloor 3 \ln(n) \rfloor$.

Locally weighted regression During the optimization process, a database, i.e., a training set is built by storing, after every evaluation on the true objective function, points together with their objective function values $(\mathbf{x}, y = f(\mathbf{x}))$. We will later then show that some points whose evaluation is asked by the optimizer are not evaluated on the true objective function. Assuming that the training set contains a sufficient number m of couples $(\mathbf{x}, f(\mathbf{x}))$, for each individual in the population, denoted now $\mathbf{q} \in \mathbb{R}^n$, locally weighted regression builds an approximate objective function using (true) evaluations stored in the training set. Kern et al [4] have tested several types of models for the objective function (linear, quadratic, ...) and have investigated the impact of the choice of the model complexity and recommend to use a full quadratic meta-model that we will hence consider in this paper. The full quadratic meta-model is built based on minimizing the following criteria w.r.t. the vector of parameters $\beta \in \mathbb{R}^{\frac{n(n+3)}{2}+1}$ of the meta-model at $\mathbf{q}$:

$$A(\mathbf{q}) = \sum_{j=1}^{m} \left[\left(\hat{f}\left(\mathbf{x}_j, \beta\right) - y_j \right)^2 K\left(\frac{d\left(\mathbf{x}_j, \mathbf{q}\right)}{h} \right) \right] \ , \tag{4}$$

where $\hat{f}$ is the meta-model defined by

$$\hat{f}\left(\mathbf{x}, \beta\right) = \beta^T \left(x_1^2, \cdots, x_n^2, \cdots, x_1 x_2, \cdots, x_{n-1} x_n, x_1, \cdots, x_n, 1 \right)^T \ , \tag{5}$$

with $\mathbf{x} = (x_1, \ldots, x_n)$. The kernel weighting function $K\left(.\right)$ is defined by $K(\zeta) = (1-\zeta^2)^2 1_{\{\zeta<1\}}$ where $1_{\{\zeta<1\}}$ is one if $\zeta < 1$ and zero otherwise, and d denotes the Mahalanobis distance with respect to the current covariance matrix $\mathbf{C}$ between 2 individuals defined as $d\left(\mathbf{x}_j, \mathbf{q}\right) = \sqrt{\left(\mathbf{x}_j - \mathbf{q}\right)^T \mathbf{C}^{-1} \left(\mathbf{x}_j - \mathbf{q}\right)}$, and h is the bandwidth defined by the distance of the kth nearest neighbor data point to $\mathbf{q}$ where $k = n\left(n+3\right) + 2$.

Table 1. Test Functions and their corresponding initial intervals and standard deviations. The starting point is uniformly drawn from the initialized interval.

Name	Function	Init.	σ^0
Noisy Sphere	$f_{\text{NSphere}}(x) = (\sum_{i=1}^{n} x_i^2) \exp(\epsilon \mathcal{N}(0,1))$	$[-3,7]^n$	5
Schwefel	$f_{\text{Schw}}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-10,10]^n$	10
Schwefel$^{1/4}$	$f_{\text{Schw}^{1/4}}(x) = (f_{\text{Schwefel}}(x))^{\frac{1}{4}}$	$[-10,10]^n$	10
Rosenbrock	$f_{\text{Rosen}}(x) = \sum_{i=1}^{n-1} \left(100.\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right)$	$[-5,5]^n$	5
Ackley	$f_{\text{Ack}}(x) = 20 - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) + e - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	$[1,30]^n$	14.5
Rastrigin	$f_{\text{Rast}}(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10.\cos(2\pi x_i))$	$[1,5]^n$	2

1 *approximate* $\hat{f}(x_k)$, $k = 1 \dots \lambda$
2 *rank* the μ best individuals ranking$_0$
3 *evaluate* f for the n_{init} best individuals, add to the training set
4 **for** $n_{ic} := 1$ **to** $\left(\frac{\lambda - n_{init}}{n_b}\right)$ **do**
5 *approximate* $\hat{f}(x_k)$, $k = 1 \dots \lambda$
6 *rank* the μ best individuals ranking$_{n_{ic}}$
7 **if** (ranking$_{n_{ic}} \neq$ ranking$_{n_{ic}-1}$) **then**
8 *evaluate* f for the n_b best unevaluated individuals, add to the training set
9 **else**
10 **break**
11 **fi**
12 **od**
13 **if** $(n_{ic} > 2)$ **then** $n_{init} = \min(n_{init} + n_b, \lambda - n_b)$
14 **elseif** $(n_{ic} < 2)$ **then** $n_{init} = \max(n_b, n_{init} - n_b)$

Fig. 1. The approximate ranking procedure, performed once the training set contains a sufficient number of evaluations to build the meta-model. n_{init} and n_b are initialized respectively to λ and max[1, $(\frac{\lambda}{10})$] .

Approximate Ranking Procedure The lmm-CMA-ES algorithm is using the approximate ranking procedure in order to decide when the quality of the model built is satisfactory [5]. This procedure heavily exploits the ranked-based property of the CMA-ES algorithm. Fig. 1 gives the implementation of this procedure proposed in [4]. Initially, a number n_{init} of best individuals based on the meta-model are evaluated using the true objective function and then added to the training set. A batch of n_b individuals is evaluated until satisfying the meta-model acceptance criterion: *keeping the ranking of each of the μ best individuals based on the meta-model unchanged for two iteration cycles.* Hence, ($n_{init,g}$ + n_{ic} n_b) individuals are evaluated every generation where n_{ic} represents the number of iteration cycles needed to satisfy the meta-model acceptance criterion. We note that n_{init} and n_b are initialized respectively to λ and max[1, $(\frac{\lambda}{10})$]. The parameter n_{init} is adapted depending on the number of iteration cycles n_{ic}: n_{init}

Table 2. Success performance SP1, i.e., the average number of function evaluations for successful runs divided by the ratio of successful runs, standard deviations of the number of function evaluations for successful runs and speedup performance spu, to reach $f_{\text{stop}} = 10^{-10}$ of lmm-CMA and nlmm-CMA. The ratio of successful runs is denoted between brackets if it is < 1.0. Results with a constant dimension $n = 5$ and an increasing λ are highlighted in grey.

Function	n	λ	ϵ	lmm-CMA		spu	nlmm-CMA		spu	CMA-ES	
f_{Rosen}	2	6		291 ± 59		2.7	**252** ± 52		3.1	779 ± 236	
	4	8		776 ± 102	[0.95]	2.8	**719** ± 54	[0.85]	3.0	2185 ± 359	[0.95]
	5	8		1131 ± 143		2.7	**1014** ± 94	[0.90]	3.0	3012 ± 394	[0.90]
	5	16		1703 ± 230	[0.95]	2.0	**901** ± 64		3.7	3319 ± 409	
	5	24		2784 ± 263		1.4	**1272** ± 90	[0.95]	3.0	3840 ± 256	
	5	32		3364 ± 221		1.3	**1567** ± 159		2.9	4515 ± 275	
	5	48		4339 ± 223		1.3	**1973** ± 144		2.9	5714 ± 297	
	5	96		6923 ± 322		1.2	**3218** ± 132		2.5	7992 ± 428	
	8	10		2545 ± 233	[0.95]	2.1	**2234** ± 202	[0.95]	2.4	5245 ± 644	
f_{Schw}	2	6		89 ± 9		4.3	87 ± 7		4.4	385 ± 35	
	4	8		166 ± 8		5.4	166 ± 6		5.4	897 ± 51	
	8	10		334 ± 9		6.2	333 ± 9		6.2	2078 ± 138	
	16	12		899 ± 40		5.9	**855** ± 30		6.2	5305 ± 166	
$f_{\text{Schw}^{1/4}}$	2	6		556 ± 25		2.4	**413** ± 25		3.3	1343 ± 72	
	4	8		1715 ± 87		1.7	**971** ± 36		2.9	2856 ± 135	
	5	8		2145 ± 69		1.6	**1302** ± 31		2.7	3522 ± 136	
	5	16		3775 ± 137		1.3	**1446** ± 31		3.4	4841 ± 127	
	5	24		5034 ± 142		1.2	**1825** ± 45		3.4	6151 ± 252	
	5	32		6397 ± 174		1.2	**2461** ± 43		3.2	7765 ± 227	
	5	48		8233 ± 190		1.2	**3150** ± 58		3.2	10178 ± 202	
	5	96		11810 ± 177		1.2	**4930** ± 94		2.9	14290 ± 252	
	8	10		4046 ± 127		1.5	**2714** ± 41		2.2	5943 ± 133	
f_{NSphere}	2	6	0.35	124 ± 14		2.7	**109** ± 12		3.1	337 ± 34	
	4	8	0.25	316 ± 45		2.3	**236** ± 19		3.1	739 ± 30	
	8	10	0.18	842 ± 77		1.8	**636** ± 33		2.4	1539 ± 69	
	16	12	0.13	2125 ± 72		1.3	2156 ± 216		1.3	2856 ± 88	
f_{Ack}	2	5		302 ± 43	[0.90]	2.6	**227** ± 23		3.5	782 ± 114	[0.95]
	5	7		1036 ± 620		2.0	**704** ± 23	[0.90]	3.0	2104 ± 117	[0.85]
	10	10		2642 ± 93	[0.90]	1.4	**2066** ± 119	[0.95]	1.8	3787 ± 151	[0.95]
f_{Rast}	2	50		898 ± 160	[0.95]	2.7	**524** ± 48	[0.95]	4.7	2440 ± 294	[0.75]
	5	70		19911 ± 599	[0.15]	0.6	**9131** ± 135	[0.15]	1.3	11676 ± 711	[0.50]
	5	140		6543 ± 569	[0.80]	1.6	**4037** ± 209	[0.60]	2.6	10338 ± 1254	[0.85]
	5	280		10851 ± 1008	[0.85]	1.3	**4949** ± 425	[0.85]	2.9	14266 ± 1069	

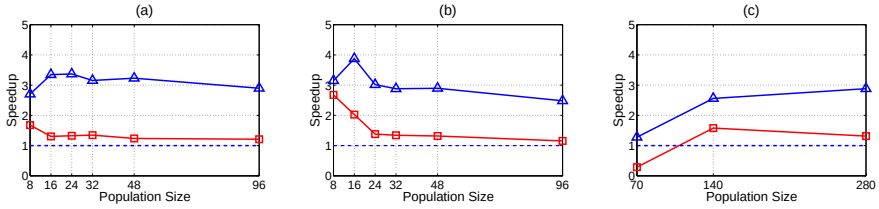

Fig. 2. Speedup of nlmm-CMA (△) and lmm-CMA (□) on (a) $f_{\text{Schw}^{1/4}}$, (b) f_{Rosen} and (c) f_{Rast} for dimension $n = 5$.

is increased if $(n_{ic} > 2)$ (Line 13 in Fig. 1) and decreased if $(n_{ic} < 2)$ (Line 14 in Fig. 1).

3 Evaluating lmm-CMA on increasing population size

3.1 Experimental procedure

The lmm-CMA and the other variants tested are evaluated on the objective functions presented in Table 1 corresponding to the functions used in [4] except two functions: (1) the function $f_{\text{Schw}^{1/4}}$ where we compose the convex quadratic

function f_{Schw} by a strictly increasing mapping $g : x \in \mathbb{R} \mapsto x^{1/4}$, introduced because we suspect that the results on f_{Schw} are artificial and only reflect the fact that the model used in lmm-CMA is quadratic and (2) the noisy sphere function f_{NSphere} whose definition has been modified following the recommendations of [6]. We have followed the experimental procedure in [4] and performed for each test function 20 independent runs using an implementation of lmm-CMA based on a java code of CMA-ES[4] randomly initialized from initial intervals defined in Table 1 and with initial standard deviations σ_0 in Table 1 and other standard parameter settings in [3]. The algorithm performance is measured using the success performance SP1 used in [7]. SP1 is defined as the average number of evaluations for successful runs divided by the ratio of successful runs, where a run is considered as successful if it succeeds in reaching $f_{\text{stop}} = 10^{-10}$. Another performance measure that might be used was the expected running time ERT [8] which is defined as the number of function evaluations conducted in all runs (successful and unsuccessful runs) divided by the ratio of successful runs. In this paper, we opt for SP1 since the stopping criteria for unsuccessful runs were not properly tuned which can affect the performance comparison. We have reproduced the results for the lmm-CMA presented in [4, Table 3]. Those results are presented in Table 2[5].

3.2 Performances of lmm-CMA with increasing population size

In lmm-CMA, a meta-model is accepted if the exact ranking of the μ best individuals remains unchanged. However, this criterion is more and more difficult to satisfy when the population size λ and thus $\mu(= \lambda/2)$ increases. We suspect that this can have drastic consequences on the performances of lmm-CMA. To test our hypothesis we perform tests for $n = 5$ on f_{Rosen}, $f_{\text{Schw}^{1/4}}$ with $\lambda = 8,\ 16,\ 24,\ 32,\ 48,\ 96$ and for f_{Rast} for $\lambda = 70,\ 140,\ 280$. The results are presented in Fig. 2 and rows highlighted in grey in Table 2. On f_{Rosen} and $f_{\text{Schw}^{1/4}}$, we observe, as expected that the speedup with respect to CMA-ES drops with increasing λ and is approaching 1. On f_{Rast}, we observe that the speedup for $\lambda = 140$ is larger than for $\lambda = 280$ (respectively equal to 1.6 and 1.3).

4 A new variant of lmm-CMA

We propose now a new variant of lmm-CMA, the new-local-meta-model CMA-ES (nlmm-CMA) that tackles the problem detected in the previous section.

4.1 A new meta-model acceptance criteria

We have seen that requiring the preservation of the exact ranking of the μ best individuals is a too conservative criterion–for population sizes larger than the

[4] See http : //www.lri.fr/∼hansen/cmaes_inmatlab.html.

[5] Experiments have been performed with $k = n(n + 3) + 2$ indicated in [4]. However we observed some differences on f_{Rosen} and $f_{\text{Schw}}$with this value of k and found out that $k = \frac{n(n+3)}{2} + 1$ allows to obtain the results presented in [4, Table 3]. We did backup this finding by using the matlab code provided by Stefan Kern.

1 *approximate* $\hat{f}(x_k)\,,\ \ k = 1 \ldots \lambda$
2 *determine* the μ best individuals set : set_0
3 *determine* the best individual : elt_0
4 *evaluate* f for the n_{init} best individuals, add to the training set
5 **for** $n_{ic} := 1$ **to** $\left(\frac{\lambda - n_{init}}{n_b}\right)$ **do**
6 *approximate* $\hat{f}(x_k)\,,\ \ k = 1 \ldots \lambda$
7 *determine* the μ best individuals set : $\mathrm{set}_{n_{ic}}$
8 *determine* the best individual : $\mathrm{elt}_{n_{ic}}$
9 **if** $(n_{init} + n_{ic}\, n_b < \frac{\lambda}{4})$
10 **if** $((\mathrm{set}_{n_{ic}} \neq \mathrm{set}_{n_{ic}-1})$ **or** $(\mathrm{elt}_{n_{ic}} \neq \mathrm{elt}_{n_{ic}-1}))$ **then**
11 *evaluate* f for the n_b best unevaluated individuals, add to the training set
12 **else**
13 **break**
14 **fi**
15 **elseif** $(\mathrm{elt}_{n_{ic}} \neq \mathrm{elt}_{n_{ic}-1})$ **then**
16 *evaluate* f for the n_b best unevaluated individuals, add to the training set
17 **else**
18 **break**
19 **fi**
20 **od**
21 **if** $(n_{ic} > 2)$ **then** $n_{init} = \min(n_{init} + n_b, \lambda - n_b)$
22 **elseif** $(n_{ic} < 2)$ **then** $n_{init} = \max(n_b, n_{init} - n_b)$

Fig. 3. The new approximate ranking procedure, performed once the training set contains a sufficient number of evaluations to build the meta-model. n_{init} and n_b are initialized respectively to λ and $\max[1, (\frac{\lambda}{10})]$.

default one–to measure the quality of meta-models. We therefore propose to replace this criterion by the following one: after building the model and ranking it, a meta-model is accepted if it succeeds in keeping, both the ensemble of μ individuals and the best individual unchanged. In this case, we ignore any change in the rank of each individual from the best μ individuals, except for the best individual which must be the same, as long as this individual is still an element of the μ best ensemble. Another criterion is added to the acceptance of the meta-model: once more than one fourth of the population is evaluated, the model is accepted if it succeeds to keep the best individual unchanged. The proposed procedure is outlined in Fig. 3. Considering only changes in the whole parent set, without taking into account the exact rank of each individual, and setting an upper limit on the number of true objective function evaluations was first proposed in [9]. The new variant is called nlmm-CMA in the sequel.

4.2 Evaluation of nlmm-CMA

The performance results of nlmm-CMA are presented in Table 2 together with the one of lmm-CMA. Table 2 shows that on f_{Rast}, the nlmm-CMA speedup is in between 2.5 and 5 instead of 1.5 and 3 for lmm-CMA and on f_{Ack}, nlmm-CMA outperforms lmm-CMA with speedups between 1.5 and 3.5 for nlmm-CMA and

Table 3. SP1, standard deviations of the number of function evaluations for successful runs and speedup performance spu, to reach $f_{\text{stop}} = 10^{-10}$ of nlmm-CMA, nlmm-CMA$_I$ (intermediate recombination and default initial parameters), nlmm-CMA$_1$(default recombination, initial values of n_{init} and n_b set to 1) and nlmm-CMA$_2$(default recombination type, $n_{init} = 1$ and $n_b = 1$ during the whole optimization process). The ratio of successful runs is denoted between brackets if it is < 1.0.

Function	n	λ	ϵ	nlmm-CMA	spu	nlmm-CMA$_I$	spu	nlmm-CMA$_1$	spu	nlmm-CMA$_2$	spu
f_{Rosen}	2	6		252 ± 52	3.1	357 ± 67	2.2	250 ± 80	3.1	229 ± 53	3.4
	4	8		719 ± 54 [0.85]	3.0	833 ± 100	2.6	596 ± 55	3.7	575 ± 68	3.8
	8	10		2234 ± 202 [0.95]	2.4	2804 ± 256 [0.95]	1.9	2122 ± 133	2.5	2466 ± 207 [0.85]	2.1
f_{Schw}	2	6		87 ± 7	4.4	110 ± 10	3.5	75 ± 8	5.2	73 ± 7	5.3
	4	8		166 ± 6	5.4	220 ± 15	4.1	138 ± 6	6.5	136 ± 5	6.6
	8	10		333 ± 9	6.2	423 ± 15	4.9	374 ± 16	5.6	380 ± 21	5.5
	16	12		855 ± 30	6.2	947 ± 24	5.6	794 ± 27	6.7	786 ± 37	6.8
$f_{\text{Schw}^{1/4}}$	2	6		413 ± 25	3.3	550 ± 29	2.4	411 ± 20	3.3	398 ± 16	3.4
	4	8		971 ± 36	2.9	1320 ± 76	2.2	938 ± 32	3.1	909 ± 30	3.1
	8	10		2714 ± 41	2.2	2714 ± 257	2.2	2668 ± 40	2.2	2677 ± 36	2.2
f_{NSphere}	2	6	.35	109 ± 12	3.1	135 ± 19	2.5	92 ± 11	3.7	87 ± 9	3.9
	4	8	.25	236 ± 19	3.1	306 ± 40	2.4	216 ± 16	3.4	219 ± 16	3.4
	8	10	.18	636 ± 33	2.4	788 ± 47	2.0	611 ± 35	2.5	619 ± 45	2.5
	16	12	.13	2156 ± 216	1.3	2690 ± 421	1.1	2161 ± 148	1.3	2195 ± 142	1.3
f_{Ack}	2	5		227 ± 23	3.5	329 ± 29 [0.85]	2.4	226 ± 21 [0.95]	3.5	208 ± 19	3.8
	5	7		704 ± 23 [0.90]	3.0	850 ± 43 [0.90]	2.5	654 ± 35 [0.95]	3.2	652 ± 32 [0.95]	3.2
	10	10		2066 ± 119 [0.95]	1.8	2159 ± 58	1.8	2394 ± 52 [0.80]	1.6	1925 ± 44	2.0
f_{Rast}	2	50		524 ± 48 [0.95]	4.7	796 ± 68 [0.75]	3.1	569 ± 26 [0.35]	4.3	1365 ± 28 [0.10]	1.8
	5	140		4037 ± 209 [0.60]	2.6	5265 ± 313 [0.55]	2.0	13685 ± 257 [0.10]	0.8	7910 ± 82 [0.10]	1.3

between 1.4 and 3 for lmm-CMA. On these functions, nlmm-CMA is significantly more efficient. For the other tested functions f_{Rast}, f_{Schw} and $f_{\text{Schw}^{1/4}}$, nlmm-CMA is marginally more efficient than the standard lmm-CMA. In Fig. 2 and highlighted rows in Table 2, we evaluate the effect of increasing λ on nlmm-CMA using the same setting as in Section 3.2. Using population sizes larger than the default one, nlmm-CMA improves CMA-ES by a factor between 2.5 and 3.5 for all tested functions f_{Rosen}, $f_{\text{Schw}^{1/4}}$ and f_{Rast}. Therefore, nlmm-CMA maintains a significant speedup for λ larger than the default one contrary to lmm-CMA which offers a speedup approaching to 1 for f_{Rosen} and $f_{\text{Schw}^{1/4}}$ and a decreasing speedup (from 1.6 to 1.3) when λ increases (from 140 to 280) for f_{Rast}.

4.3 Impact of the recombination type

The choice of the recombination type has an important impact on the efficiency of ES in general [10] and CMA-ES in particular [2, 3]. In the previous section, all the runs performed use the default weighted recombination type defined by Eq. 3. In the new variant of lmm-CMA, the meta-model acceptance criterion does not take into account the exact rank of each individual except the best one. By modifying the meta-model acceptance criteria of lmm-CMA, a possible accepted meta-model may be a meta-model that preserves the μ best individuals set and the best individual but generates a ranking far from the "true" ranking, i.e., the one based on the true objective function. We now compare nlmm-CMA using weighted recombination where weights are defined in Eq. 3 and intermediate recombination where weights are all equal to $1/\mu$: nlmm-CMA$_I$. Results are presented in Table 3. The algorithm nlmm-CMA outperforms nlmm-CMA$_I$ in all cases suggesting that even if the exact ranking is not taken into account for assessing the quality of the meta-model in nlmm-CMA , this ranking is not random and has still an amount of information to guide CMA-ES.

4.4 Impact of initial parameters

In the tests presented so far, the initial parameters of the approximate ranking procedure ($n_{init},\ n_b$) were initialized at the beginning of the optimization process to (λ, max[1, ($\frac{\lambda}{10}$)]). Every generation g, the number of initial individuals evaluated n_{init} is adapted (increased or decreased) depending on the meta-model quality (Lines 21 and 22 in Fig 3). The number of evaluations performed every generation is ($n_{init,g} + n_{ic,g} \times n_b$). We quantify now the impact of the initial values of (n_{init} and n_b) on the total cost of the optimization process. The algorithm nlmm-CMA is compared to a similar version where initial parameters are chosen as small as possible, i.e., n_{init} and n_b are equal to 1. Moreover, we consider two cases: (1) with update denoted nlmm-CMA$_1$, i.e., where initial parameters are adapted depending on the iteration cycle number (Lines 21 and 22 in Fig 3), and (2) without update denoted nlmm-CMA$_2$, i.e., parameters are equal to 1 during the entire optimization process (omitting lines 21 and 22 in Fig. 3). We note that in case (1), the number of evaluations for each generation g is ($n_{init,g} + n_{ic,g} \times n_b$) where $n_{init,0} = 1$ and $n_b = 1$. In case (2), every generation, lmm-CMA evaluates ($1 + n_{ic,g}$) individuals. The results on different test functions are summarized in Table 3.

On the unimodal functions f_{Schw}, $f_{\text{Schw}^{1/4}}$, setting n_{init} and n_b as small as possible in every generation, is marginally more efficient than the default definition of initial parameters on small dimensions except for dimension $n = 8$ and $\lambda = 10$. On f_{Rosen}, nlmm-CMA$_2$ is the most efficient compared to other approaches, except for dimension $n = 8$ and $\lambda = 10$ which can be justified by a higher number of unsuccessful runs compared to other approaches. On the multimodal function f_{Ack}, modifying the initial parameter n_{init} does not have an important impact on the speedup of lmm-CMA (between 1.5 and 4). However on f_{Rast}, using a small initial n_{init} decreases considerably the probability of success of the optimization, from 0.95 to between 0.35 and 0.10 for dimension $n = 2$ and $\lambda = 50$, and from 0.60 to 0.10 for dimension $n = 5$ and $\lambda = 140$. Evaluating only an individual every iteration cycle does not cause, in general, any changes in the ranking of the μ best individuals and therefore it causes a premature acceptance of the meta-model. Hence, with a small initial parameters, the optimization process can easily get stuck in local minima. However, setting a sufficiently large initial n_{init} and a batch size n_b proportional to λ prevents the algorithm from a premature acceptance of the meta-model. These results confirm the initial parameters choice suggested in [4].

5 Summary

In this work, we have investigated the performances of the lmm-CMA algorithm coupling CMA-ES with local meta-models. On f_{Rosen} and $f_{\text{Schw}^{1/4}}$, we have shown that the speedup of lmm-CMA with respect to CMA-ES drops to one when the population size λ increases. This phenomenon has been explained by the too restrictive condition used to stop evaluating new points dedicated at refining the meta-model, namely requiring that the *exact* ranking of the $\mu = \lambda/2$

best solutions is preserved when evaluating a new solution on the exact objective function. To tackle this problem, we have proposed to relax the condition to: *the set of μ best solutions* is preserved and *the best individual* is preserved. The resulting new variant, nlmm-CMA outperforms lmm-CMA on the test functions investigated and the speedup with CMA-ES is between 1.5 and 7. Moreover, contrary to lmm-CMA it maintains a significant speedup, between 2.5 and 4, when increasing λ on f_{Rosen}, $f_{\text{Schw}^{1/4}}$ and f_{Rast}. The study of the impact of the recombination weights has shown that the default weights of CMA-ES are more appropriate than equal weights. The influence of two parameters, n_b and n_{init}, corresponding to the number of individuals evaluated respectively initially and in each iteration cycle has been investigated. We have seen that setting those parameters to 1 during the whole optimization process can marginally improve the performances on uni-modal functions and some multimodal test functions. However it increases the likelihood to be stuck in local minima for the Rastrigin function suggesting that the default parameter for lmm-CMA are still a good choice for nlmm-CMA.

Acknowledgments The authors would like to thank Nikolaus Hansen for insightful discussions. This work was partially funded by the French National Research Agency (ANR) grant No. ANR-08-COSI-007.

References

1. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.
2. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
3. N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *PPSN'04*. Springer Verlag, 2004.
4. S. Kern, N. Hansen, and P. Koumoutsakos. Local meta-models for optimization using evolution strategies. In *Parallel Problem Solving from Nature PPSN X*, pages 939–948. Springer, 2006.
5. T. P. Runarsson. Constrained evolutionary optimization by approximate ranking and surrogate models. In *Parallel Problem Solving from Nature PPSN VIII*, LNCS 3242, pages 401–408. Springer, 2004.
6. M. Jebalia, A. Auger, and N. Hansen. Log linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, 2010. accepted.
7. A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.
8. N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Research Report RR-6828, INRIA, 2009.
9. T. P. Runarsson. Approximate evolution strategy using stochastic ranking. In *IEEE Congress on Evolutionary Computation*, pages 745–752, 2006.
10. D. V. Arnold. Optimal weighted recombination. In *Foundations of Genetic Algorithms 8*, pages 215–237. Springer Verlag, 2005.