Local-Meta-Model CMA-ES for Partially Separable Functions
Zyed Bouzarkouna, Anne Auger, Didier Yu Ding

To cite this version:

HAL Id: hal-00588977
https://hal.archives-ouvertes.fr/hal-00588977v2
Submitted on 2 May 2011
Local-Meta-Model CMA-ES for Partially Separable Functions

Zyed Bouzarkouna1, 2, Anne Auger2, Didier Yu Ding1
zyed.bouzarkouna@ifpen.fr, anne.auger@inria.fr, d-yu.ding@ifpen.fr
1 IFP Energies nouvelles
92852 Rueil-Malmaison, France
2 TAO Team, INRIA Saclay-Ile-de-France
LRI, Paris-Sud University
91405 Orsay, France

ABSTRACT
In this paper, we propose a new variant of the covariance matrix adaptation evolution strategy with local meta-models (lmm-CMA) for optimizing partially separable functions. We propose to exploit partial separability by building at each iteration a meta-model for each element function (or sub-function) using a full quadratic local model. After introducing the approach we present some first experiments using element functions with dimensions 2 and 4. Our results demonstrate that, as expected, exploiting partial separability leads to an important speedup compared to the standard CMA-ES. We show on the tested functions that the speedup increases with increasing dimensions for a fixed dimension of the element function. On the standard Rosenbrock function the maximum speedup of $\lambda$ is reached in dimension 40 using element functions of dimension 2. We show also that higher speedups can be achieved by increasing the population size. The choice of the number of points used to build the meta-model is also described and the computational cost is discussed.

Categories and Subject Descriptors
G.1.6 [Optimization]: Global Optimization; I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic Methods

General Terms
Algorithms

Keywords
Optimization, Partial separability, Problem structure, Covariance matrix adaptation, Evolution strategy, CMA-ES, Meta-models

1. INTRODUCTION
Many real-world engineering problems involve the optimization of expensive objective functions. An introductory example is the optimization of the production scheme of an oil and gas field, where the objective function implies a fluid flow simulation which involves solving a large number of linear systems of equations and requires an evaluation time of several minutes to several hours [3].

In this context, it is worth trying to exploit the maximum information possible on the problem—even if it increases consequently the internal cost of the optimization procedure—in order to save some expensive function evaluations and reduce the overall computational time of the optimization procedure. A classical approach to do that is to model the objective function by a so-called metamodel or surrogate function and save some evaluations by evaluating some solutions on the meta-model instead of the true expensive function.

In the realm of continuous optimization, the covariance matrix adaptation evolution strategy (CMA-ES) [11, 10], a population based stochastic search algorithm, is recognized as one of the most powerful derivative-free optimization methods [9]. The CMA-ES exhibits many invariances, a desirable property as it implies the generalization of results from one function to a class of functions and confer thus robustness and wider applicability of the method. In particular, CMA-ES is a rank-based search algorithm exploiting the objective function only through the relative ranking of solutions within the population. The rank-based property implies invariance of the algorithm on the class of functions class $f = \{g \circ f, \ g : \mathbb{R} \rightarrow \mathbb{R} \text{ strictly increasing}\}$ for any $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Though invariances are desirable, assuming that the search procedure one is willing to improve—which will be for us CMA-ES—exploits optimally the maximal information on the optimization problems while keeping invariances implies that we need to exploit more information on the problem to optimize than the original algorithm in order to obtain consequent improvements. This thus leads inevitably to breaking some invariances of the original algorithm.

The CMA-ES algorithm has been combined with local meta-models that are constructed using points from the archive of solutions—called the training set—evaluated on the (expensive) original objective function. The quality of the meta-models is appraised using an approximate ranking procedure that determines if the objective function predicted by the meta-model is good enough or more points should be evaluated on the original function. The resulting al-
algorithm is called lmm-CMA [12]. The original acceptance criterion for the meta-models proposed for lmm-CMA has been shown to be too conservative for increasing population sizes and modified in order to maintain a reasonable speed-up when population sizes larger than the default one are used [2]. In lmm-CMA, the underlying model is locally quadratic, i.e., the objective function is modeled as a convex-quadratic function and thus the algorithm exploits that some functions can be locally approximated by quadratic models. Consequently, invariance to rank-preserving transformations is lost [2, 13] explaining on the other hand the improvements that are observed on functions that can be locally accurately modeled as quadratic functions.

With the motivation to preserve the invariance to monotonic transformations, rank-based Support Vector Machine were introduced as surrogate models within CMA-ES in the ACM-ES algorithm [13]. Speedups with respect to CMA-ES between 0.5 and 4.5 were observed on seven uni- and multimodal benchmark functions, with dimension in-between 2 and 40. However, unfortunately, the improvements observed lack yet of a reasonable explanation as the method is not presented as exploiting further information on the objective function than the original CMA-ES.

In this paper we investigate the optimization of the class of partially separable functions where a function of n-variables is partially separable if it is the sum of a number of so-called element functions each depending on a smaller number of variables.

Exploiting partial separability or separability is a common approach to enhance performances of optimization algorithms, in particular when dealing with large scale optimization. For instance a trust region algorithm for monotonic transformations, rank-based Support Vector Machine were introduced as surrogate models within CMA-ES in the lmm-CMA by building meta-models as a sum of element functions depending on fewer variables, only know in general that the problem can be decomposed as a restricted knowledge about the structure of the problem.

This particular case of partially separable function is considered for instance in [1, 5, 7]. A more general definition, given in [14], considers that each sub-function can depend on a number of variables that are a linear combination of a subset of variables.

In this paper we consider a generalization of the previous definitions allowing non-linear combinations of the subset of variables. More precisely a function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is said partially separable if there exists an integer \( N > 1 \), a set of integers \((n_i)_{1 \leq i \leq N}\) with \( n_i < n \), for all \( i = 1, \ldots, N \), a set of explicit functions \( \Phi : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_i} \) and a set of functions \( f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R} \), such that \( f \) can be written as \( f(x) = \sum_{i=1}^{N} f_i(\Phi_i(x)) \). The sub-functions or element functions \( f_i \) depend on a number \( n_i \) of parameters called element variables. The functions \( \Phi_i \) will be called mapping functions. Note that the setting of [14] is recovered by taking \( \Phi = U' \) where \( U' \) is a linear mapping from \( \mathbb{R}^{n_i} \) to \( \mathbb{R}^{n_i} \).

For a given partially separable function, there exists “theoretically” an infinite number of ways to define the element functions and mapping functions. However, one has usually a restricted knowledge about the structure of the problem that determines the modeling choice. We can argue that we only know in general that the problem can be decomposed as a sum of element functions depending on fewer variables, and that there is thus no reason to encode non-linearity in the variable dependencies. However, a motivating example for our general definition is the following. We consider the problem one has to face in petroleum engineering industry of finding the optimal well configurations that one is willing to drill. One has to find optimal locations of several wells to maximize the profit of the production scheme deployed on a considered field which is defined as the sum of the profits related to each well drilled. A suitable way to model the objective function is to suppose that the profit corresponding to a given well depends only on its location and on the distances of this well to the others. Using the distances between the wells as an element variable implies using a non-linear combination of the parameters of the problem.

In this same well placement problem, the objective function is computed using a numerical software able to simulate for a given set of well placements the quantity of oil, water and gas that can be extracted from each well. Consequently one has access to the function value of each element function. As argued above this assumption is reasonable as it models the case for the well placement problem [4]. History matching is another problem in petroleum engineering in which this assumption is reasonable. In history matching problems, we want to adjust the reservoir model until it closely reproduces the past

\[ f(x) = \sum_{i=1}^{n-1} h(x_i, x_{i+1}) \]
behavior of the reservoir (historical production and pressures). For this problem also, we can define the objective function as a sum of a number of sub-functions defined for each well and calculated when evaluating the objective function [6].

3. CMA-ES WITH LOCAL META-MODELS

In this section we present the basics about CMA-ES [11, 10] and hnnm-CMA a variant of CMA-ES with local meta-models [2].

3.1 CMA-ES

The CMA-ES algorithm [11, 10] is an evolution strategy in which at each generation g, a population of λ points \((x_i^{(g)}, \ i = 1, \ldots, \lambda)\) is sampled according to a multivariate normal distribution:

\[ x_i^{(g)} = m_i^{(g)} + \sigma^{(g)} N_i(0, C^{(g)}) \]

where \((m_i^{(g)}, g \in N)\) defines the sequence of mean values of the multivariate normal distribution generated by CMA-ES, constituting the sequence of estimate of the optimum, \(\{\sigma_i^{(g)}, g \in N\}\) denotes the sequences of step-sizes, and \(\{C_i^{(g)}, g \in N\}\) the sequences of covariance matrices.

The random vectors \(N_i(0, C^{(g)})\) are independent multivariate normal distributions with zero mean vector and covariance matrix \(C^{(g)}\). Those λ individuals are ranked according to \(f\):

\[ f(x_{i+1}^{(g)}) \leq \ldots \leq f(x_{\mu}^{(g)}) \leq \ldots \leq f(x_{\lambda}^{(g)}) \]

where we use the notation \(x_j^{(g)}\) for \(i\)th best individual. The mean \(m_i^{(g)}\) is then updated by taking the weighted mean of the best \(\mu\) individuals:

\[ m_j^{(g+1)} = \mu \sum_{i=1}^{\mu} \omega_i x_j^{(g)} \]

where in general \(\mu = \frac{1}{4}\) and \((\omega_i)_{1 \leq i \leq \mu}\) are strictly positive and normalized weights.

The covariance matrix of the search distribution \(C^{(g)}\) is updated after evaluation of the population, to the local shape of the fitness landscape. Furthermore, the step-size \(\sigma^{(g)}\) is updated as well after evaluation of the population. The update equations for \(m^{(g)}\), \(C^{(g)}\) and \(\sigma^{(g)}\) are given in Algorithm 1 taking as input the vector \((x_1^{(g)}, \ldots, x_{\lambda}^{(g)})\) and where dependency in \(g\) is omitted. We also refer to [10] for those update equations and for the setting of default parameters.

3.2 Building the meta-model

To build an approximate model denoted by \(\hat{f}\) of the objective function \(f\), we use a locally weighted regression. During the optimization, a training set is built by storing, after every evaluation on the true objective function, points together with their objective function values. Assuming that the training set contains a sufficient number \(m\) of elements, let us consider an individual, denoted now by \(q \in \mathbb{R}^n\), to be evaluated with the approximate model. We select from the training set the \(k\) nearest points \((x_j, j = 1, \ldots, k)\) to \(q\) using the Mahalanobis distance \(d\) with respect to the current covariance matrix \(C\). The distance between \(q\) and a point \(z \in \mathbb{R}^n\) is then given by \(d(z,q) = \sqrt{(z-q)^T C^{-1} (z-q)}\).

Locally weighted regression builds an approximate model using the (true) evaluations \((y_j, j = 1, \ldots, k)\) stored in the training set and corresponding to the \(k\) selected nearest points to \(q\). The use of a full quadratic meta-model is suggested in [12]. Hence, using a vector \(\beta \in \mathbb{R}^{\frac{n(n+1)}{2}+1}\), we define \(\hat{f}\) for a given point \(z = (z_1, \ldots, z_n)\) as follows:

\[ \hat{f}(z,\beta) = \beta^T (z_1^2, \ldots, z_n^2, z_1z_2, \ldots, z_{n-1}z_n, z_1, \ldots, z_n)^T. \]  

The full quadratic meta-model is built based on minimizing the following criterion w.r.t. the vector of parameters \(\beta\) of the meta-model at \(q\):

\[ A(q) = \sum_{j=1}^{k} \left( (f(x_j,\beta) - y_j) \right)^2 K \left( \frac{d(x_j,q)}{h} \right) \]

The kernel weighting function \(K(\cdot)\) is defined by \(K(\cdot) = 1 - \frac{\cdot^2}{C^2}\) where \(C = 1\) if \(\cdot < 1\) and zero otherwise, and \(h\) is the bandwidth defined by the distance of the \(k\)th nearest neighbor data point to \(q\). In order to build the full quadratic meta-model, \(k\) must be greater or equal to \(k_{\text{min}} = \frac{n(n+1)}{2} + 1\). The setting \(k = n(n+3)/2\) is suggested in [12]. The sufficient size of the training set denoted above by \(m\) must be then greater or equal to \(k\).

3.3 CMA-ES with meta-models

The \(n\)hnnm-CMA algorithm combines the CMA-ES with local meta-models by exploiting the fact that the updates of CMA-ES only rely on the ranking of the \(\mu\) best solutions. An iteration of hnnm-CMA consists of one iteration of CMA-ES where the evaluation step on \(f\) that usually determines the ranking of the \(\mu\) best solutions is replaced by the approximate ranking procedure that outputs an approximate ranking of the candidate solutions and that costs maximally \(\lambda\) function evaluations on \(f\). The mean value, covariance matrix and step-size of CMA-ES are then updated according to Algorithm 1 using this approximate ranking.

The approximate ranking procedure works as follows. If the training set contains enough points to build the meta-model, for each individual proposed by CMA-ES, one meta-model is computed according to the procedure described in Section 3.2. The best \(n_{\text{init}}\) individuals according to the meta-models are evaluated on \(f\) and then added to the training set, and for each non-evaluated individual a meta-model is built anew. If the acceptance criterion given below is not satisfied, we loop over the following steps 1) the \(n_b\) best individuals (according to the meta-models) unevaluated individuals are evaluated on \(f\) and added to the training set, 2) meta-models are rebuilt for the unevaluated individuals and the acceptance criterion is checked. The loop breaks when either the acceptance criterion is satisfied or the \(\lambda\) individuals are evaluated. The acceptance criterion defining hnnm-CMA reads:

- the best individual and the ensemble of \(\mu\) best individuals remain unchanged, if less than one fourth of the population is evaluated;
- the best individual remains unchanged, if more than one fourth of the population is evaluated.

Hence, \(n_{\text{init}} + n_b + n_{\text{inc}}\) individuals are evaluated every generation where \(n_{\text{inc}}\) represents the number of iteration cycles needed to satisfy the meta-model acceptance criterion. The integer \(n_b\) is chosen to be equal to \(\max[1, \left(\frac{3}{10}\right)]\) and \(n_{\text{init}}\) is...
initialized to $\lambda$ and adapted after every generation depending on $n_c$ according to:

$$
n_{\text{init}} \leftarrow \min(n_{\text{init}} + n_c, \lambda - n_h), \quad \text{if } (n_{\text{inc}} > 2),$$

$$
n_{\text{init}} \leftarrow \max(n_{\text{init}} - n_h), \quad \text{if } (n_{\text{inc}} < 2).
$$

(7)

The minimum number of evaluations performed for a given generation, which corresponds to the minimum value that $n_{\text{init}}$ can reach, is then equal to $n_h$. We refer to [2, Fig. 3] for a detailed outline of the procedure.

4. LMM-CMA FOR PARTIALLY SEPARABLE FUNCTIONS

This section introduces a new algorithm based on nlmm-CMA and exploiting the partial separability of the objective function. This algorithm will be called p-sep lmm-CMA.

In our proposed approach, the partial separability of the objective function is exploited when building the meta-models. The optimization process defined by CMA-ES is not altered. The idea behind exploiting the problem structure when building the meta-model, is to improve the quality of the approximate model. Hence, the better the quality of the model is, the easier the acceptance criteria can be satisfied, the less evaluations are performed.

Let us consider a partially separable function $f$. As in Section 2, we consider that $f$ has $N$ element functions $(f_i)_{1 \leq i \leq N}$. For each element function, we associate a mapping function $\Phi^i$ such that $f(x) = \sum_{i=1}^{N} f_i \circ \Phi^i(x)$. We suppose that when evaluating a point $x$ on $f$, we have access to the evaluations $(f_i \circ \Phi^i(x))_{1 \leq i \leq N}$ as well.

In Section 3.2, an approximate function $\hat{f}$ for a given objective function $f$ is defined using a locally weighted regression based on the training set containing both evaluated points and their values on $f$. In this section, we propose to build a meta-model for each element function $f_i$ that we denote by $\hat{f}_i$. The meta-model $\hat{f}$ of $f$ is then defined by

$$
\hat{f} = \sum_{i=1}^{N} \hat{f}_i \circ \Phi^i.
$$

(8)

The meta-model $\hat{f}_i$ of each element function $f_i$ is built in a way quite similar to the meta-model $\hat{f}$ of $f$ defined in Section 3.2. The training set is built by storing for every evaluated point $x$, $\Phi^i(x)$ and its corresponding values on $f_i$, i.e. $f_i(\Phi^i(x))$. Let us consider an individual $q$ for which $\Phi^i(q) \in \mathbb{R}^n_i$. In Section 3.2, the idea behind the training set contains a sufficient number $m_i$ of elements, we select the $k_i \in \mathbb{N}$ nearest points $(\Phi(x_j)), j = 1, \ldots, k_i$, to $\Phi(q)$ using the Mahalanobis distance, with respect to a matrix $C_i$, defined for a given point $\mathbf{z} \in \mathbb{R}^n$ as

$$
d_i(\Phi(z), \Phi(q)) = (\Phi(z) - \Phi(q))^T C_i^{-1} (\Phi(z) - \Phi(q)).
$$

(9)

where $C_i$ is an $n_i \times n_i$ matrix adapted to the local shape of the landscape of $f_i$ (see below).

Similarly to Section 3.2, a full quadratic meta-model is used. Using a vector $\beta_i \in \mathbb{R}^{n_i(n_i+1)/2}$, $\hat{f}_i$ is defined for a given point $\mathbf{z} \in \mathbb{R}^n$, for which we denote $\Phi^i(z) = (\mathbf{z}_1, \ldots, \mathbf{z}_{n_i})$ as

$$
\hat{f}_i(\Phi^i(z), \beta_i) = \mathbf{z}_i^T \mathbf{z}_i^T \beta_i.
$$

(10)

where $\mathbf{z}_i = (u_{i1}^2, \ldots, u_{i_{n_i}}^2, \ldots, u_{n_{i-1}n_i}, u_{n_{i+1}}, \ldots, u_{n_i})$. The full quadratic meta-model is built by minimizing the following criterion with respect to $\beta_i$:

$$
B(q) = \sum_{i=1}^{k_i} \left( \hat{f}_i(\Phi^i(x_j), \beta_i) - f_i(\Phi^i(x_j)) \right)^2 
\times K \left( \frac{d_i(\Phi^i(x_j), \Phi^i(q))}{h} \right).
$$

(11)

where $K(.)$ is the kernel weighting function defined as in Section 3.2, and $h$ is the bandwidth defined by the distance $d_i$ of the $k_i$th nearest neighbor data point to $q$. For a given element function, $k_i$ must be greater or equal to $k_{i, \text{min}} = 2m_i(n_i+1) + 1$. $k_i$ is chosen to be equal to $2 \times k_{i, \text{min}}$. The choice of $k_i$ will be discussed in Sec. 5.3. The sufficient size of the training set denoted above by $m_i$ must be then greater or equal to $k_i$.

Hence, the approximate function of $f$ which corresponds to $\hat{f}(x) = \sum_{i=1}^{N} \hat{f}_i(\Phi^i(x))$ is incorporated into CMA-ES using the approximate ranking procedure as detailed in Section 3.3.

It remains now to describe how the matrices $(C_i)_{1 \leq i \leq N}$ are obtained. They are built in an iterative manner. At each iteration, after the approximate ranking procedure, each of the $\lambda$ candidate solutions denoted $(X_m)_{1 \leq m \leq \lambda}$ and sampled according to (2) has been either evaluated on $f$ or has associated approximate meta-models value given by (8). Thus for each $i$, the vectors $\Phi^i(X_m) \in \mathbb{R}^{n_i}$ have either been evaluated on $f_i$ or have an associated estimate of $f_i$ provided by $\hat{f}_i$. We then consider the vectors $\Phi^i(X_m) \in \mathbb{R}^{n_i}$ for $1 \leq m \leq \lambda$ and rank them according to $\hat{f}_i$, where $\hat{f}_i$ equals $f_i$ if $X_m$ was evaluated on $f$ and $f_i$ otherwise. The ordered $\mu$ best solutions according to $\hat{f}_i$ are used as input variables in Algorithm 1, to update the covariance matrix $C_i$.

In Algorithm 1, the parameters $(\omega), \epsilon_{\text{stop}}, c_{\text{stop}}, c_{\text{cov}}, \mu_{\text{cov}}, d_{\sigma}$ are chosen with default values as defined in [10]. Initial values for $p_e$, $p_c$ and $C$ used in Algorithm 1 are also set to default as in [10]. Initial values for $m$ and $\sigma$ are set to $\Phi^i(\mathbf{m}^{(0)})$ and $\sigma^{(0)}$ where $\mathbf{m}^{(0)}$ and $\sigma^{(0)}$ are the initial mean vector and step-size of (a)lmm-CMA. The idea behind this adaptation procedure is the same as the one of the adaptive encoding proposed in [8]. However in adaptive encoding, step-size update is not needed and different normalizations for the weights depending on the step-length are introduced. Though we believe that the adaptive encoding update is more robust numerically, it has not been tested for this work.

5. EVALUATION OF P-SEP LMM-CMA

In this section we describe the functions used to evaluate p-sep lmm-CMA. We show the performance of this method compared to CMA-ES. The optimal bandwidth used to build the meta-model is also investigated and the computational cost of the approach is discussed.

5.1 Test functions

The p-sep lmm-CMA is evaluated on the partially separable test functions $f_{\text{Rosen}}$, $f_{\text{Rosen}}$, $f_{\text{Rosen}}$, $f_{\text{Rosen}}$, $f^{\text{BlockEff}}$ defined in Table 1. For the block-rotated ellipsoid, $Q$ is a $2 \times 2$ rotation matrix sampled uniformly anew for every run performed. The performance of the method is measured using the success performance SP1 defined as the average number of evaluations for successful runs divided
**Table 1:** Test functions. For the block-rotated ellipsoid, \( Q \) is a 2 \times 2 rotation matrix with each column being a uniformly distributed unit vector.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Number of evaluations</th>
<th>( n = 4 )</th>
<th>( n = 8 )</th>
<th>( n = 10 )</th>
<th>( n = 16 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock</td>
<td>( f_{\text{Rosen}}(x) = \sum_{i=1}^{n-1} \left( \alpha_i (x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right) )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td></td>
</tr>
<tr>
<td>Rosenbrock ( \frac{1}{2} )</td>
<td>( f_{\text{Rosen}}^{\frac{1}{2}}(x) = \sum_{i=1}^{n-1} \left( \alpha_i (x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right)^{\frac{1}{2}} )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td></td>
</tr>
<tr>
<td>Block-rotated ellipsoid 2D</td>
<td>( f_{\text{BlockElli-2D}}(x, y) = \frac{2}{n} \sum_{i=1}^{n} \left( \alpha_i (x_i - 1)^2 + (y_i - 1)^2 \right) )</td>
<td>( f_1(\mathbf{u}) = f_{\text{BlockElli-2D}}(u_1, u_2) )</td>
<td>( f_1(\mathbf{u}) = f_{\text{BlockElli-2D}}(u_1, u_2) )</td>
<td>( f_1(\mathbf{u}) = f_{\text{BlockElli-2D}}(u_1, u_2) )</td>
<td>( f_1(\mathbf{u}) = f_{\text{BlockElli-2D}}(u_1, u_2) )</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2:** Modeling of the partially separable functions tested.

<table>
<thead>
<tr>
<th>Name</th>
<th>( n_M )</th>
<th>( N )</th>
<th>( f_1(\mathbf{u}) = f_{\text{Rosen}}(\mathbf{u}) )</th>
<th>( \Phi(\mathbf{v} = (v_i)) = (v_i, v_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenbrock</td>
<td>2 ((n-1))</td>
<td>( (n-1) )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( \Phi(\mathbf{v}) = (v_i, v_i) )</td>
</tr>
<tr>
<td>Rosenbrock ( \frac{1}{2} )</td>
<td>2 ((n-1))</td>
<td>( (n-1) )</td>
<td>( f_1(\mathbf{u}) = \alpha_i (u_i^2 - u_{i+1})^2 + (u_i - 1)^2 )</td>
<td>( \Phi(\mathbf{v}) = (v_i, v_i) )</td>
</tr>
<tr>
<td>Block-rotated ellipsoid</td>
<td>2 ((n-1))</td>
<td>( (n-1) )</td>
<td>( f_1(\mathbf{u}) = f_{\text{BlockElli-2D}}(u_1, u_2) )</td>
<td>( \Phi(\mathbf{v}) = (v_1, v_2) )</td>
</tr>
</tbody>
</table>

**Algorithm 1:** CMA-Update(\( x_1, \ldots, x_n \))

1. given parameters \( (\omega_i)_{1 \leq i \leq n_M} \), \( c_{e}, c_c, c_{\text{cov}}, \mu_{\text{cov}}, d_{e} \).
2. Set \( \mu_{\text{eff}} = 1/\sum_{i=1}^{n} \omega_i^2 \).
3. \( \mathbf{m} \leftarrow \mathbf{m} \).
4. \( \mathbf{m} \leftarrow \sum_{i=1}^{n} \omega_i x_i \).
5. \( \mathbf{p}_e \leftarrow (1 - c_{e}) \mathbf{p}_e + \sqrt{c_{e}(2 - c_{e}) \mu_{\text{eff}}} \mathbf{C}^{-\frac{1}{2}} \mathbf{m} \mathbf{m}^T \).
6. \( \mathbf{p}_e \leftarrow (1 - c_{e}) \mathbf{p}_e + \sqrt{c_{e}(2 - c_{e}) \mu_{\text{eff}}} \mathbf{m} \mathbf{m}^T \).
7. \( \mathbf{C} \leftarrow \sum_{i=1}^{n} \omega_i (x_i - m)(x_i - m)^T \).
8. \( \sigma \leftarrow \sqrt{\sigma^2 + \frac{1}{\mu_{\text{cov}}} \mathbf{p}_e \mathbf{p}_e^T + c_{\text{cov}} (1 - \frac{1}{\mu_{\text{cov}}}) \mathbf{C} \mu} \).
9. \( \mathbf{C} \leftarrow (1 - c_{\text{cov}}) \mathbf{C} + \frac{1}{\mu_{\text{cov}}} \mathbf{p}_e \mathbf{p}_e^T + c_{\text{cov}} (1 - \frac{1}{\mu_{\text{cov}}}) \mathbf{C} \mu \).

by the ratio of successful runs, needed to reach a stopping objective value \( f_{\text{stop}} = 10^{-10} \), except for \( f_{\text{Rosen}}^{\frac{1}{2}} \) for which \( f_{\text{stop}} = 10^{-5} \). We perform 20 independent runs to measure SP1. The runs are randomly initialized in the intervals \([-5, 5]\) for \( f_{\text{Rosen}}, f_{\text{Rosen}}^{100}, f_{\text{Rosen}}^{10000}, f_{\text{BlockElli}} \) and \([-10, 10]\) for \( f_{\text{BlockElli}} \). Each test function is modeled by defining a number \( N \) of element functions, a number \( n_M \) of element variables for each element function, a set of element functions denoted by \( f_i : \mathbb{R}^{n_M} \rightarrow \mathbb{R} \) and a set of mapping functions \( \Phi_i : \mathbb{R}^n \rightarrow \mathbb{R}^{n_M} \), such that \( f = \sum_{i=1}^{N} f_i \circ \Phi_i \). The modeling of each test function is shown in Table 2. The block-rotated ellipsoid function is defined using quadratic element functions. For the other tested functions, the defined element functions are not quadratic.
5.2 Performance of p-sep lmm-CMA

Results on the test functions are presented in Table 3 showing the performance of p-sep lmm-CMA compared to CMA-ES and to some tests with nlmm-CMA. For each test, by defining the value of $n_M$, we refer to the corresponding modeling defined in Table 2. It is clear that exploiting the partial separability within CMA-ES with meta-models improves the performance of CMA-ES with a speedup in between 4.5 and 15.

For element functions with fixed $n_M$ equal to 2, p-sep lmm-CMA offers an increasing speedup with increasing dimensions of the problem as shown in Fig. 2. The algorithm p-sep lmm-CMA performs better with increasing dimensions since it breaks the curse of dimensionality when building the meta-model: for a problem of dimension $n$, building the meta-model is equivalent to building $N$ meta-models of dimension $n_M$.

Using greater number of parameters for each separated meta-model decreases the speedup obtained by the approach. On $f^\text{Rosen}_{100}$ for a dimension 16, the speedup, decreases from 8.6 to 2.2 for corresponding values of $n_M$ respectively equal to 2 and 4.

At each iteration at least $n_b$ function evaluations are performed on the true function in order to check the accuracy of the meta-models. The parameter $n_b$ is set to $\max\{1, \left\lfloor \frac{1}{4} \right\rfloor \}$. This setting is introduced in order to be able to add a significant amount of information at each iteration by enriching the training set. It is in particular important when dealing with large population sizes. For increasing population sizes $\lambda$, i.e., for increasing values of $\mu$, we need an increasing number of points evaluated at each iteration cycle to be able to have a significant impact on the ranking of population.

Moreover, a better setting of $n_M$ would also depend on the dimension of the problem as for increasing dimensions, i.e., for increasing numbers $k$ (or $k_i$) of points to build the meta-model, we need an increasing number of points evaluated at each iteration cycle to be able to change significantly the meta-model and then the ranking of the population.

The minimum number of evaluations performed at each iteration $n_b$ limits the speedup that can be achieved by our approach. We show that for some test functions, we are able to reach this maximum speedup of 20, we reach a speedup of 995, 7. This setting is introduced in order to be able to add a significant amount of information at each iteration by enriching the training set. It is in particular important when dealing with large population sizes. For increasing population sizes $\lambda$, i.e., for increasing values of $\mu$, we need an increasing number of points evaluated at each iteration cycle to be able to have a significant impact on the ranking of population. Furthermore, a better setting of $n_M$ would also depend on the dimension of the problem as for increasing dimensions, i.e., for increasing numbers $k$ (or $k_i$) of points to build the meta-model, we need an increasing number of points evaluated at each iteration cycle to be able to change significantly the meta-model and then the ranking of the population.

The minimum number of evaluations performed at each iteration $n_b$ limits the speedup that can be achieved by our approach. We show that for some test functions, we are able to reach this maximum speedup of $\lambda/n_M$. For $f^\text{Rosen}_{100}$ with $n = 40$ and for $f^\text{Rosen}_4$ with $n = 20$, we reach a speedup equal to $\lambda$ since $n_b$ is equal to 1 in these tests.

Since we reach the maximal speedup allowed by the approach on the Rosenbrock function, we asked ourselves

<table>
<thead>
<tr>
<th>Function</th>
<th>n</th>
<th>$n_M$</th>
<th>$\lambda$</th>
<th>p-sep lmm-CMA</th>
<th>CMA-ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f^\text{Rosen}_{100}$</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>180 ± 13</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>308 ± 20</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>353 ± 20</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
<td>12</td>
<td>465 ± 20</td>
<td>8.6</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2</td>
<td>12</td>
<td>548 ± 34</td>
<td>9.1</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2</td>
<td>14</td>
<td>755 ± 32</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2</td>
<td>15</td>
<td>871 ± 41</td>
<td>11.2</td>
</tr>
<tr>
<td>$f^\text{Rosen}_{100/2}$</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>1333 ± 238</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>2745 ± 246</td>
<td>6.6</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>5552 ± 429</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
<td>12</td>
<td>10583 ± 398</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2</td>
<td>12</td>
<td>14749 ± 431</td>
<td>6.3</td>
</tr>
<tr>
<td>$f^\text{BlockEll}_{100}$</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>544 ± 48</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>1008 ± 67</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>1299 ± 178</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
<td>12</td>
<td>3346 ± 223</td>
<td>9.9</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2</td>
<td>12</td>
<td>6797 ± 878</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2</td>
<td>14</td>
<td>20751 ± 2116</td>
<td>8.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>n</th>
<th>$n_M$</th>
<th>$\lambda$</th>
<th>p-sep lmm-CMA</th>
<th>CMA-ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f^\text{Rosen}_{100}$</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>226 ± 11</td>
<td>6.6</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>392 ± 14</td>
<td>8.2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>472 ± 17</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2</td>
<td>12</td>
<td>670 ± 37</td>
<td>9.8</td>
</tr>
</tbody>
</table>
whether we can further reduce the number of overall function evaluations needed to reach a target by increasing the population size $\lambda$. The default population size denoted $\lambda_{\text{default}}$ value equals $4 + \lceil 3 \times \ln(n) \rceil$. Fig. 1(a) shows the influence of the population size on the performance of p-sep lmm-CMA.

We perform 20 independent runs on $f_{Rosen}$ for dimensions $n = 4, 8, 10$ and 16, and $n_M = 2$ with $f_{\text{stop}} = 10^{-10}$. The tested population sizes are written as $\lambda = \gamma \times \lambda_{\text{default}}$ where $\gamma$ is in-between 1 and 10. Tests were performed with similar parameters: $n_{\text{init}}$ initialized to $\lambda_{\text{default}}$ and $n_i$ equal to $\max[1, \lceil \frac{\lambda_{\text{default}}}{\gamma} \rceil]$. A training set containing $k_i$ elements randomly sampled is loaded at the beginning of every run in order to use the meta-models from the first generation, for all the tests. Results show that $\lambda = 4 \times \lambda_{\text{default}}$ gives the minimum number of evaluations to reach $f_{\text{stop}}$ and improves the performance by a factor between 1.5 and 2 over the default population size. For $\gamma > 4$, the performance of p-sep lmm-CMA stagnates. We observe in Fig. 1(b) that the number of evaluations per generation increases linearly for increasing population sizes.

5.3 Optimal bandwidth for building partially separated meta-models

Let us consider an element function $f_i$ with a number of element variables $n_i$. The optimal bandwidth depends on the number of points $k_i$ used to build the meta-model. As shown in Section 4, $k_i$ must be greater or equal to $k_{i,\text{min}} = \lceil \frac{n_i(n_i+3)}{2} \rceil + 1$. In this section, we investigate the influence of the choice of $k_i$ on the performance of p-sep lmm-CMA. We perform 20 independent runs on $f_{Rosen}^\alpha$ for $\alpha = 1, 10^2, 10^4$ and $f_{Rosen}^{100}$ for different dimensions in-between 4 and 40. Results are shown in Fig. 3, where $k_i$ is written as $k_i = \beta \times k_{i,\text{min}}$ for $\beta = 1, 2, 3, 4$ and 5. We find that for 14 tests over the 23 tests performed on the test functions with different dimensions, a good estimate of the optimal $\beta$ is equal to 2. Moreover, for the other tests, choosing a value of $\beta$ equal to 2 is a reasonable choice since it offers a speedup close to best one found, except for $f_{Rosen}^{100}$ with dimensions 10 and 16.

5.4 Computational cost

The internal cost of the optimization procedure is dominated by the evaluation of the objective function and the construction of the meta-model.

For p-sep lmm-CMA, building a meta-model consists in finding in the training set the $k_i$ sorted nearest points to the point to be evaluated and then solving (11). Let us consider a training set with a size $m$. To find and sort the best $k_i$ points, we begin by sorting the first $k_i$ points of the training set using a heapsort algorithm which has a complexity of $k_i \log k_i$. Then, we compare the other ($m - k_i$) points with the selected $k_i$ points until finding its position which adds at worst a complexity of $(m - k_i) \times k_i$. Thus, finding and sorting the best $k_i$ points needs $O(k_i \log k_i + (m - k_i)k_i) = O(m \times k_i)$. According to Sec. 5.3, the optimal bandwidth $k_i$ is equal to $n_i(n_i+3)+2$. Thus, finding and sorting the points to evaluate the meta-model needs $O(m \times n_i^2)$. Moreover, solving (11) is dominated by a $k_i \times k_i$ matrix inversion and thus has a complexity of $n_i^6$.

Let us denote by $N_e$ the number of evaluations on the true objective function and by $N_m$ the number of built meta-models. The complexity of p-sep lmm-CMA is then equal to: $N_e + N_m n_i^2 (m + n_i^2)$.

6. CONCLUSIONS

In this paper we have investigated the exploitation of partial separability of the objective function to enhance the performances of CMA-ES coupled with local meta-models. We have defined p-sep lmm-CMA, a new variant of CMA-ES with meta-models for partially separable functions. In this variant, we build separate meta-models for each element function, instead of building one meta-model for the whole objective function. We have shown that the speedup of p-sep lmm-CMA with respect to CMA-ES is in-between 4.5
and 15 for the tested functions. For $f_{\text{Rosen}}^{100}$, with a dimension 40 and for $f_{\text{Rosen}}^{100}$, with a dimension 20, we reach a speedup equal to $\lambda$ which corresponds to the theoretical maximum speedup allowed by the approach. In general, the maximum speedup that can be achieved equals $\lambda/n_b$ as at least $n_b$ evaluations on the true function are performed at each iteration. We have shown through preliminary tests on the standard Rosenbrock function that increasing the population size allows to decrease significantly (by a factor between 1.5 and 2) the number of evaluations to reach a given target. The optimal population size on the Rosenbrock function is shown to be equal to $4 \times \lambda_{\text{default}}$. Future investigations are needed to define the optimal population size depending on the dimension of the problem and the dimension of the sub-problems, over a wide range of test functions.

Acknowledgments
The authors would like to thank Nikolaus Hansen for numerous helpful discussions. This work was partially funded by the ANR-08-COSI-007-12 grant and by the FUI of System@tic Paris-Region ICT cluster through contract DGT 117 407 Complex Systems Design Lab (CSDL).

7. REFERENCES