Local-Meta-Model CMA-ES for Partially Separable Functions

Zyed Bouzarkouna, Anne Auger, Didier Yu Ding

Local-Meta-Model CMA-ES for Partially Separable Functions

Zyed Bouzarkouna[1, 2]
zyed.bouzarkouna@ifpen.fr

Anne Auger[2]
anne.auger@inria.fr

Didier Yu Ding[1]
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 λ 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

GECCO'11, July 12–16, 2011, Dublin, Ireland.

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 $\text{class}\, f = \{g \circ f,\ g : \mathbb{R} \to \mathbb{R} \text{ strictly increasing}\}$ for any $f : \mathbb{R}^n \to \mathbb{R}$. 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-

gorithm 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 minimizing partially separable functions was proposed in [5]. Separability was also exploited within CMA-ES. A method where the covariance matrix was constrained to be diagonal has been proposed in [15].

In this paper, we propose to exploit partial separability within the lmm-CMA by building meta-models as a sum of meta-models depending on a smaller number of variables. As we will discuss in the next section, we will assume that we can exploit the function value of the element functions and not only the function value of the overall function. This setting is motivated by the well placement problem in which the numerical simulations can output the production of each single well though the objective is to maximize the production of all the wells.

This paper is structured as follows. Section 2 defines a general notion of partial separability. Section 3 recalls the combination of CMA-ES with meta-models proposed in [2]. In Section 4, we propose a new variant of CMA-ES with meta-models for partially separable functions. The performance of this variant is evaluated in Section 5 on a number of partially separable test functions. The choice of the number of points used to build the meta-model is also described and the computational cost is discussed.

2. PARTIAL SEPARABILITY AND PROBLEM MODELING

A function $f : \mathbb{R}^n \to \mathbb{R}$ is partially separable if it can be written as a sum of sub-functions, also called *element functions*, each depending on a fewer number of variables. Often the particular case where each sub-function depends on a subset of variables of the original function is defined as partial separability. For instance the Rosenbrock function in Table 1 writes

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} h(x_i, x_{i+1}) \ , \tag{1}$$

where $\mathbf{x} = (x_i)_{1\leq i\leq n}$ and $h(x_i, x_{i+1}) = \alpha(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$ and is thus partially separable with each sub-function depending on the subset of variables $[(x_i, x_{i+1})]_{i=1,\ldots,n-1}$. 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 $f : \mathbb{R}^n \to \mathbb{R}$ 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 $(\Phi^i : \mathbb{R}^n \to \mathbb{R}^{n_i})_{1\leq i\leq N}$ and a set of functions $(f_i : \mathbb{R}^{n_i} \to \mathbb{R})_{1\leq i\leq N}$, such that f can be written as $f(\mathbf{x}) = \sum_{i=1}^{N} f_i(\Phi^i(\mathbf{x}))$. The sub-functions or element functions $(f_i)_{1\leq i\leq N}$ depend on a number n_i of parameters called *element variables*. The functions Φ^i will be called *mapping functions*. Note that the setting of [14] is recovered by taking $\Phi^i = U^i$ where U^i is a linear mapping from $\mathbb{R}^n$ 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 nonlinear combination of the parameters of the problem [4].

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*. In the following we will also assume not only that the function is partially separable but also that 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

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 nlmm-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 $(\mathbf{x}_i^{(g)},\ i = 1, \ldots, \lambda)$ is sampled according to a multivariate normal distribution:

$$\mathbf{x}_i^{(g)} = \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}_i(0, \mathbf{C}^{(g)}), \qquad \text{for } i = 1, \ldots, \lambda \ , \quad (2)$$

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

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

$$f(\mathbf{x}_{1:\lambda}^{(g)}) \leq \ldots \leq f(\mathbf{x}_{\mu:\lambda}^{(g)}) \leq \ldots \leq f(\mathbf{x}_{\lambda:\lambda}^{(g)}) \ , \quad (3)$$

where we use the notation $\mathbf{x}_{i:\lambda}^{(g)}$ for i^{th} best individual. The mean $\mathbf{m}^{(g)}$ is then updated by taking the weighted mean of the best μ individuals:

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} \omega_i \mathbf{x}_{i:\lambda}^{(g)} \ , \quad (4)$$

where in general $\mu = \frac{\lambda}{2}$ and $(w_i)_{1 \leq i \leq \mu}$ are strictly positive and normalized weights.

The covariance matrix of the search distribution $\mathbf{C}^{(g)}$ 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 $\mathbf{m}^{(g)}$, $\mathbf{C}^{(g)}$ and $\sigma^{(g)}$ are given in Algorithm 1 taking as input the vector $(\mathbf{x}_{1:\lambda}^{(g)}, \ldots, \mathbf{x}_{\mu:\lambda}^{(g)})$ 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 $\mathbf{q} \in \mathbb{R}^n$, to be evaluated with the approximate model. We select from the training set the $k \in \mathbb{N}$ nearest points $(\mathbf{x}_j,\ j = 1, \ldots, k)$ to $\mathbf{q}$ using the Mahalanobis distance d with respect to the current covariance matrix $\mathbf{C}$. The distance between $\mathbf{q}$ and a point $\mathbf{z} \in \mathbb{R}^n$ is then given by $d(\mathbf{z}, \mathbf{q}) = \sqrt{(\mathbf{z} - \mathbf{q})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{q})}$.

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 $\mathbf{q}$. The use of a full quadratic meta-model is suggested in [12]. Hence, using a vector $\beta \in \mathbb{R}^{\frac{n(n+3)}{2}+1}$, we define $\hat{f}$ for a given point $\mathbf{z} = (z_1, \ldots, z_n)$ as follows

$$\hat{f}(\mathbf{z}, \beta) = \beta^T \big(z_1^2, \cdots, z_n^2, z_1 z_2, \cdots, z_{n-1} z_n, z_1, \cdots, z_n, 1\big)^T. \quad (5)$$

The full quadratic meta-model is built based on minimizing the following criterion w.r.t. the vector of parameters β of the meta-model at $\mathbf{q}$:

$$A(\mathbf{q}) = \sum_{j=1}^{k} \left[\left(\hat{f}(\mathbf{x}_j, \beta) - y_j \right)^2 K\left(\frac{d(\mathbf{x}_j, \mathbf{q})}{h} \right) \right] \ . \quad (6)$$

The kernel weighting function $K(.)$ 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 h is the bandwidth defined by the distance of the k^{th} nearest neighbor data point to $\mathbf{q}$. In order to build the full quadratic meta-model, k must be greater or equal to $k_{\min} = \frac{n(n+3)}{2} + 1$. 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)lmm-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 μ best solutions. An iteration of lmm-CMA consists of one iteration of CMA-ES where the evaluation step on f that usually determines the ranking of the μ best solutions is replaced by the approximate ranking procedure that outputs an approximate ranking of the candidate solutions and that costs maximally λ 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_{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 (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 λ individuals are evaluated. The acceptance criterion defining nlmm-CMA reads:

- the *best individual* and the *ensemble of μ 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, $(n_{\text{init}} + n_{\text{b}} * n_{\text{ic}})$ individuals are evaluated every generation where n_{ic} 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, $(\frac{\lambda}{10})$] and n_{init} is

initialized to λ and adapted after every generation depending on n_{ic} according to:

$$\begin{array}{ll} n_{\text{init}} \leftarrow \min(n_{\text{init}} + n_{\text{b}}, \lambda - n_{\text{b}}), & \text{if}(n_{\text{ic}} > 2) \ , \\ n_{\text{init}} \leftarrow \max(n_{\text{b}}, n_{\text{init}} - n_{\text{b}}), & \text{if}(n_{\text{ic}} < 2) \ . \end{array} \quad (7)$$

The minimum number of evaluations performed for a given generation, which corresponds to the minimum value that n_{init} can reach, is then equal to n_b. 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 Φ^i such that $f(\mathbf{x}) = \sum_{i=1}^{N} f_i \circ \Phi^i(\mathbf{x})$. We suppose that when evaluating a point $\mathbf{x}$ on f, we have access to the evaluations $(f_i \circ \Phi^i(\mathbf{x}))_{1\leq i\leq N}$ 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

$$\hat{f} = \sum_{i=1}^{N} \hat{f}_i \circ \Phi^i \ . \quad (8)$$

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 $\mathbf{x}$, $\Phi^i(\mathbf{x})$ and its corresponding values on f_i, i.e. $f_i(\Phi^i(\mathbf{x}))$. Let us consider an individual $\mathbf{q}$ for which $\Phi^i(\mathbf{q}) \in \mathbb{R}^{n_i}$ has to be evaluated on the approximate model of f_i. Assuming that the training set contains a sufficient number m_i of elements, we select the $k_i \in \mathbb{N}$ nearest points $(\Phi^i(\mathbf{x}_j), j = 1, \ldots, k_i)$ to $\Phi^i(\mathbf{q})$ using the Mahalanobis distance d_i with respect to a matrix $\mathbf{C}_i$, defined for a given point $\mathbf{z} \in \mathbb{R}^n$ as

$$d_i(\Phi^i(\mathbf{z}), \Phi^i(\mathbf{q})) = \sqrt{(\Phi^i(\mathbf{z}) - \Phi^i(\mathbf{q}))^T {\mathbf{C}_i}^{-1} (\Phi^i(\mathbf{z}) - \Phi^i(\mathbf{q}))}, \quad (9)$$

where $\mathbf{C}_i$ 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 $\beta_i \in \mathbb{R}^{\frac{n_i(n_i+3)}{2}+1}$, $\hat{f}_i$ is defined for a given point $\mathbf{z} \in \mathbb{R}^n$, for which we denote $\Phi^i(\mathbf{z}) = (\tilde{u}_1, \ldots, \tilde{u}_{n_i})$ as

$$\hat{f}_i \left(\Phi^i(\mathbf{z}), \beta_i \right) = \beta_i^T \tilde{\mathbf{z}}_i^T \ , \quad (10)$$

where $\tilde{\mathbf{z}}_i = \left(\tilde{u}_1^2, \cdots, \tilde{u}_{n_i}^2, \tilde{u}_1\tilde{u}_2, \cdots, \tilde{u}_{n_i-1}\tilde{u}_{n_i}, \tilde{u}_1, \cdots, \tilde{u}_{n_i}, 1\right)$. The full quadratic meta-model is built by minimizing the following criterion with resepct to β_i:

$$B(\mathbf{q}) = \sum_{j=1}^{k_i} \left[\left(\hat{f}_i \left(\Phi^i(\mathbf{x}_j), \beta_i \right) - f_i(\Phi^i(\mathbf{x}_j)) \right)^2 \times K \left(\frac{d_i\left(\Phi^i(\mathbf{x}_j), \Phi^i(\mathbf{q})\right)}{h} \right) \right] \ . \quad (11)$$

$K(.)$ 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 $\mathbf{q}$. For a given element function, k_i must be greater or equal to $k_{i,\min} = \frac{n_i(n_i+3)}{2}+1$. k_i is chosen to be equal to $2 \times k_{i,\min}$. 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 $\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \hat{f}_i(\Phi^i(\mathbf{x}))$ is incorporated into CMA-ES using the approximate ranking procedure as detailed in Section 3.3.

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

In Algorithm 1, the parameters $(\omega_i)_{1\leq i\leq \mu}$, c_σ, c_{c}, c_{cov}, μ_{cov}, d_σ are chosen with default values as defined in [10]. Initial values for $\mathbf{p}_\sigma$, $\mathbf{p}_c$ and $\mathbf{C}$ used in Algorithm 1 are also set to default as in [10]. Initial values for $\mathbf{m}$ and σ 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 (n)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^1_{Rosen}, f^{100}_{Rosen}, f^{10000}_{Rosen}, $f^{100}_{\text{Rosen}\,\frac{1}{2}}$ and $f_{\text{BlockElli}}$ defined in Table 1. For the block-rotated ellipsoid, $\mathbf{Q}$ is a 2×2 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, $\mathbf{Q}$ is a 2×2 rotation matrix with each column being a uniformly distributed unit vector.

Name	Function
Rosenbrock	$f^{\alpha}_{\text{Rosen}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(\alpha.\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right)$
Rosenbrock$^{\frac{1}{2}}$	$f^{\alpha}_{\text{Rosen}^{\frac{1}{2}}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(\alpha.\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right)^{\frac{1}{2}}$
Block-rotated ellipsoid 2D	$f^{\alpha}_{\text{BlockElli}-\text{2D}}(x, y) = \sum_{i=1}^{2} \left(\alpha^{\frac{i-1}{n-1}}.(\mathbf{Q} \times (x, y))^2\right)$
Block-rotated ellipsoid	$f^{\alpha}_{\text{BlockElli}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(f^{\alpha}_{\text{BlockElli}-\text{2D}}(x_i, x_{i+1})\right)$

Table 2: Modeling of the partially separable functions tested.

Name	n_{M}	N	$f_i(\mathbf{u} = (u_j)_{1 \le j \le n_{\text{M}}})$			$\Phi_i(\mathbf{v} = (v_j)_{1 \le j \le n})$
Rosenbrock	2	$(n-1)$	$f_i(\mathbf{u})$	$=$	$\alpha.\left(u_1^2 - u_2\right)^2 + (u_1 - 1)^2$	$\Phi_i(\mathbf{v}) = (v_i, v_{i+1})$
	4	$\frac{n-1}{3}$	$f_i(\mathbf{u})$	$=$	$\alpha.\left(u_1^2 - u_2\right)^2 + (u_1 - 1)^2$ $+\alpha.\left(u_2^2 - u_3\right)^2 + (u_2 - 1)^2$ $+\alpha.\left(u_3^2 - u_4\right)^2 + (u_3 - 1)^2$	$\Phi_i(\mathbf{v}) = (v_{3i-2}, v_{3i-1}, v_{3i}, v_{3i+1})$
Rosenbrock$^{\frac{1}{2}}$	2	$(n-1)$	$f_i(\mathbf{u})$	$=$	$\left(\alpha.\left(u_1^2 - u_2\right)^2 + (u_1 - 1)^2\right)^{\frac{1}{2}}$	$\Phi_i(\mathbf{v}) = (v_i, v_{i+1})$
Block-rotated ellipsoid	2	$(n-1)$	$f_i(\mathbf{u})$	$=$	$f^{\alpha}_{\text{BlockElli}-\text{2D}}(u_1, u_2)$	$\Phi_i(\mathbf{v}) = (v_{2i-1}, v_{2i})$

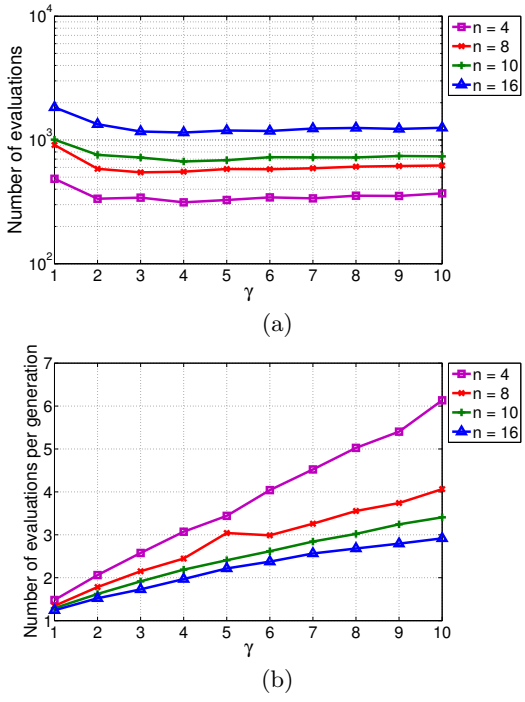


Figure 1: (a) Average number of evaluations of the p-sep lmm-CMA on f^{100}_{Rosen} to reach f_{stop} for varying population sizes $\lambda = \gamma \times \lambda_{\text{default}}$. (b) Average number of evaluations per generation of the p-sep lmm-CMA on f^{100}_{Rosen} for varying population sizes $\lambda = \gamma \times \lambda_{\text{default}}$.

Algorithm 1: CMA-Update$(\mathbf{x}_1, \ldots, \mathbf{x}_\mu)$

1. given parameters $(\omega_i)_{1 \le i \le \mu}$, c_σ, c_{c}, c_{cov}, μ_{cov}, d_σ. Set $\mu_{\text{eff}} = 1/\sum_{i=1}^{\mu} {\omega_i}^2$
2. given $\mathbf{m} \in \mathbb{R}^n$, $\mathbf{p}_\sigma \in \mathbb{R}^n$, $\mathbf{p}_c \in \mathbb{R}^n$, $\sigma \in \mathbb{R}$ and $\mathbf{C} \in \mathbb{R}^{n \times n}$ from last iteration
3. $\mathbf{m}^- \leftarrow \mathbf{m}$
4. $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} \omega_i \mathbf{x}_i$
5. $\mathbf{p}_\sigma \leftarrow (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{-\frac{1}{2}} \frac{\mathbf{m} - \mathbf{m}^-}{\sigma}$
6. $\mathbf{p}_c \leftarrow (1 - c_{\text{c}})\mathbf{p}_c + \sqrt{c_{\text{c}}(2 - c_{\text{c}})\mu_{\text{eff}}} \frac{\mathbf{m} - \mathbf{m}^-}{\sigma}$
7. $\mathbf{C}_\mu = \sum_{i=1}^{\mu} \omega_i \frac{(\mathbf{x}_i - \mathbf{m}^-)(\mathbf{x}_i - \mathbf{m}^-)^{\text{T}}}{\sigma^2}$
8. $\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\text{E}\|\mathcal{N}(0, \mathbf{I})\|} - 1\right)\right)$
9. $\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\mathbf{p}_{\text{c}}\mathbf{p}_{\text{c}}^{\text{T}} + c_{\text{cov}}\left(1 - \frac{1}{\mu_{\text{cov}}}\right) \times \mathbf{C}_\mu$

by the ratio of successful runs, needed to reach a stopping objective value $f_{\text{stop}} = 10^{-10}$, except for $f^{\alpha}_{\text{Rosen}^{\frac{1}{2}}}$ 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^{1}_{Rosen}, f^{100}_{Rosen}, f^{10000}_{Rosen} and $f^{10000}_{\text{Rosen}^{\frac{1}{2}}}$ 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 $f_i : \mathbb{R}^{n_{\text{M}}} \to \mathbb{R}$ and a set of mapping functions $\Phi_i : \mathbb{R}^n \to \mathbb{R}^{n_{\text{M}}}$, 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.

Table 3: 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 p-sep lmm-CMA, nlmm-CMA and CMA-ES (for $f^{100}_{\text{Rosen}\frac{1}{2}}$, $f_{\text{stop}} = 10^{-5}$). The ratio of successful runs is denoted between brackets if it is < 1.0. The number of element variables of each element function is denoted by n_{M}.

Function	n	n_{M}	λ	p-sep lmm-CMA			spu	nlmm-CMA			spu	CMA-ES		
f^{1}_{Rosen}	4	2	8	**189**	± 13		5.1	297	± 20		3.2	964	± 192	
	8	2	10	**308**	± 20		6.5	932	± 52		2.2	2006	± 118	
	10	2	10	**353**	± 20		6.8	1482	± 169		1.6	2418	± 204	
	16	2	12	**465**	± 20		8.6					4023	± 310	
	20	2	12	**548**	± 34		9.1					4978	± 374	
	32	2	14	**755**	± 32		10.3					7777	± 347	
	40	2	15	**871**	± 41		11.2					9799	± 602	
f^{100}_{Rosen}	4	2	8	**485**	± 47	[0.80]	4.7	647	± 67	[0.95]	3.5	2269	± 254	[0.85]
	8	2	10	**910**	± 71	[0.80]	6.5	2602	± 264	[0.85]	2.3	5883	± 727	[0.90]
	10	2	10	**1006**	± 99	[0.95]	7.6	3727	± 300	[0.90]	2.1	7644	± 765	[0.95]
	16	2	12	**1834**	± 117	[0.90]	8.6					15781	± 1360	[0.85]
	16	4	12	**7162**	± 1112	[0.95]	2.2					15781	± 1360	[0.85]
	20	2	12	**2533**	± 361	[0.90]	10.4					26366	± 3249	[0.85]
	32	2	14	**4628**	± 144	[0.95]	13.2					60948	± 2668	[0.90]
	40	2	15	**6527**	± 226	[0.95]	15.2					99346	± 3502	[0.85]
f^{10000}_{Rosen}	4	2	8	**1333**	± 238	[0.95]	5.3	2637	± 715	[0.90]	2.7	7032	± 944	[0.90]
	8	2	10	**2745**	± 246		6.6	10287	± 468	[0.85]	1.8	18216	± 1683	[0.95]
	10	2	10	**5552**	± 429	[0.75]	4.5	16280	± 843	[0.85]	1.5	25037	± 3160	[0.95]
	16	2	12	**10583**	± 398	[0.80]	5.9					62903	± 4441	[0.90]
	20	2	12	**14749**	± 431	[0.90]	6.3					93545	± 6566	[0.95]
$f^{100}_{\text{Rosen}\frac{1}{2}}$	4	2	8	**544**	± 48	[0.70]	4.8	909	± 75	[0.75]	2.9	2620	± 342	[0.95]
	8	2	10	**1008**	± 67	[0.80]	7.0	2549	± 262	[0.95]	2.8	7006	± 762	
	10	2	10	**1299**	± 178	[0.95]	10.4	4685	± 518	[0.90]	2.9	13517	± 1288	[0.75]
	16	2	12	**3346**	± 223	[0.90]	9.9					33154	± 3568	[0.90]
	20	2	12	**6797**	± 878	[0.85]	10.0					68136	± 5363	[0.80]
	32	2	14	**20751**	± 2116	[0.85]	14.6					302039	± 40915	[0.65]
$f^{10000}_{\text{BlockElli}}$	4	2	8	**226**	± 11		6.6					1500	± 89	
	8	2	10	**392**	± 14		8.2					3220	± 196	
	10	2	10	**472**	± 17		8.7					4093	± 173	
	16	2	12	**670**	± 37		9.8					6566	± 284	

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}.

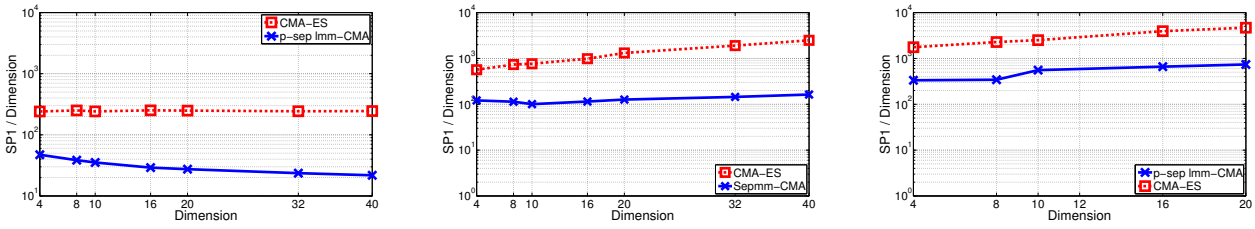
Using greater number of parameters for each separated meta-model decreases the speedup obtained by the approach. On f^{100}_{Rosen} for a dimension 16, the speedup, decreases from 8.6 to 2.2 for corresponding values of n_{M} respectively equal to 2 and 4.

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 $\max[1, (\frac{\lambda}{10})]$. 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 λ, i.e., for increasing values of μ, 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_b 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.

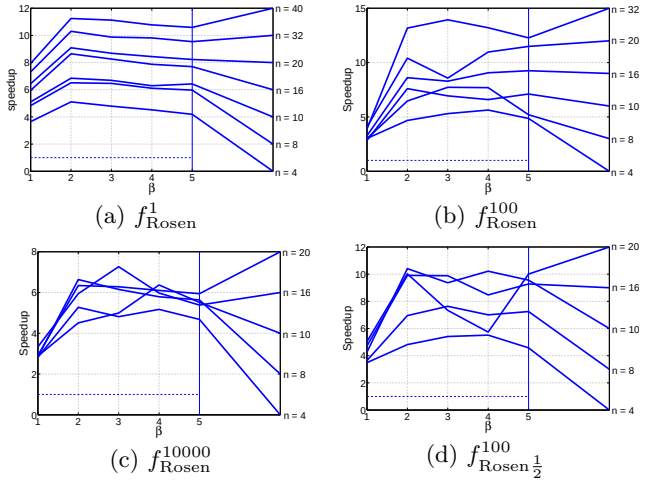
The minimum number of evaluations performed at each iteration n_{b} limits the speedup that can be achieved by our approach. We show that for some test functions, we are able to reach this maximum speedup of λ/n_{b}. For f^{100}_{Rosen} with $n = 40$ and for $f^{100}_{\text{Rosen}\frac{1}{2}}$ with $n = 20$, we reach a speedup equal to λ 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



(a) f^{1}_{Rosen} (b) f^{100}_{Rosen} (c) f^{10000}_{Rosen}

Figure 2: Success performance SP1 over the dimension of the problem on $f^{\alpha}_{\text{Rosen}}$, with $\alpha = 1$, 10^2 and 10^4 for dimensions in between 4 and 40. The dimension of the sub-functions n_{M} equals 2.



(a) f^{1}_{Rosen} (b) f^{100}_{Rosen}

(c) f^{10000}_{Rosen} (d) $f^{100}_{\text{Rosen}\,\frac{1}{2}}$

Figure 3: Average speedup with respect to CMA-ES to reach f_{stop} with a varying number of points used to build the meta-model $k_i = \beta \times k_{i,\min}$ where $k_{i,\min} = \frac{n_i(n_i+3)}{2} + 1$. Each point corresponds to 20 runs performed.

whether we can further reduce the number of overall function evaluations needed to reach a target by increasing the population size λ. The default population size denoted λ_{default} value equals $4 + \lfloor 3 \times \ln(n) \rfloor$. 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^{100}_{Rosen} for dimensions $n = 4$, 8, 10 and 16, and $n_{\text{M}} = 2$ with $f_{\text{stop}} = 10^{-10}$. The tested population sizes are written as $\lambda = \gamma \times \lambda_{\text{default}}$ where γ is in-between 1 and 10. Tests were performed with similar parameters: n_{init} initialized to λ_{default} and n_{b} equal to $\max[1, (\frac{\lambda_{\text{default}}}{10})]$. 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_{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 $k_{i,\min} = \frac{n_i(n_i+3)}{2} + 1$. 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 $f^{\alpha}_{\text{Rosen}}$ for $\alpha = 1$, 10^2, 10^4 and $f^{100}_{\text{Rosen}\,\frac{1}{2}}$ for different dimensions in-between 4 and 40. Results are shown in Fig. 3, where k_i is written as $k_i = \beta \times k_{i,\min}$ 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 β is equal to 2. Moreover, for the other tests, choosing a value of β equal to 2 is a reasonable choice since it offers a speedup close to best one found, except for f^{100}_{Rosen} 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: $N_e + N_m \; n_i^2(m + n_i^4)$.

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_{Rosen}^{100} with a dimension 40 and for $f_{\text{Rosen}\frac{1}{2}}^{100}$ with a dimension 20, we reach a speedup equal to λ which corresponds to the theoretical maximum speedup allowed by the approach. In general, the maximum speedup that can be achieved equals λ/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

[1] A. Bouaricha and J. Morè. Impact of partial separability on large-scale optimization. *Computational Optimization and Applications*, 7:27–40, 1997.

[2] Z. Bouzarkouna, A. Auger, and D. Ding. Investigating the local-meta-model CMA-ES for large population sizes. In C. Di Chio et al., editors, *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 402–411. Springer Berlin / Heidelberg, 2010.

[3] Z. Bouzarkouna, D. Ding, and A. Auger. Using evolution strategy with meta-models for well placement optimization. In *12th European Conference on the Mathematics of Oil Recovery (ECMOR XII)*. EAGE, 2010.

[4] Z. Bouzarkouna, D. Ding, and A. Auger. Partially separated meta-models with evolution strategies for well placement optimization. In *SPE EUROPEC/EAGE annual conference and exhibition*, number SPE 143292, May 2011.

[5] B. Colson and P. L. Toint. Optimizing partially separable functions without derivatives. *Optimization Methods and Software*, 20(4-5):493–508, 2005.

[6] D. Ding and F. Mckee. Using partial separability of the objective function for gradient-based optimizations in history matching. In *SPE reservoir simulation symposium*, number SPE 140811, February 2011.

[7] N. Durand and J.-M. Alliot. Genetic crossover operator for partially separable functions. In *Proceedings of the third annual Genetic Programming Conference*, 1998.

[8] N. Hansen. Adaptive encoding: How to render search coordinate system invariant. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature - PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 205–214. Springer Berlin / Heidelberg, 2008.

[9] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *GECCO '10: Proceedings of the 12th annual conference comp on Genetic and evolutionary computation*, pages 1689–1696, New York, NY, USA, 2010. ACM.

[10] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 282–291. Springer Berlin / Heidelberg, 2004.

[11] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[12] S. Kern, N. Hansen, and P. Koumoutsakos. Local meta-models for optimization using evolution strategies. In T. Runarsson et al., editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 939–948. Springer Berlin / Heidelberg, 2006.

[13] I. Loshchilov, M. Schoenauer, and M. Sebag. Comparison-based optimizers need comparison-based surrogates. In R. Schaefer et al., editors, *Parallel Problem Solving from Nature - PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin / Heidelberg, 2011.

[14] J. Nocedal. Large scale unconstrained optimization. In *The state of the art in numerical analysis*, pages 311–338. Oxford University Press, 1996.

[15] R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*, pages 296–305. Springer, 2008.