

Benchmarking the Local Metamodel CMA-ES on the Noiseless BBOB'2013 Test Bed

Anne Auger, Dimo Brockhoff, Nikolaus Hansen

▶ **To cite this version:**

Anne Auger, Dimo Brockhoff, Nikolaus Hansen. Benchmarking the Local Metamodel CMA-ES on the Noiseless BBOB'2013 Test Bed. GECCO (Companion), workshop on Black-Box Optimization Benchmarking (BBOB'2013), Jul 2013, Amsterdam, Netherlands. hal-00825840

HAL Id: hal-00825840

https://inria.hal.science/hal-00825840

Submitted on 24 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Benchmarking the Local Metamodel CMA-ES on the Noiseless BBOB'2013 Test Bed

Anne Auger
Projet TAO, INRIA
Saclay—Ile-de-France
LRI, Bât 660, Univ. Paris-Sud
91405 Orsay Cedex, France
lastname@lri.fr

Dimo Brockhoff
INRIA Lille - Nord Europe
Dolphin team
59650 Villeneuve d'Ascq
France
firstname.lastname@inria.fr

Nikolaus Hansen
Projet TAO, INRIA
Saclay—Ile-de-France
LRI, Bât 660, Univ. Paris-Sud
91405 Orsay Cedex, France
lastname@lri.fr

ABSTRACT

This paper evaluates the performance of a variant of the local-meta-model CMA-ES (lmm-CMA) in the BBOB 2013 expensive setting. The lmm-CMA is a surrogate variant of the CMA-ES algorithm. Function evaluations are saved by building, with weighted regression, full quadratic meta-models to estimate the candidate solutions' function values. The quality of the approximation is appraised by checking how much the predicted rank changes when evaluating a fraction of the candidate solutions on the original objective function. The results are compared with the CMA-ES without meta-modeling and with previously benchmarked algorithms, namely BFGS, NEWUOA and saACM.

It turns out that the additional meta-modeling improves the performance of CMA-ES on almost all BBOB functions while giving significantly worse results only on the attractive sector function. Over all functions, the performance is comparable with saACM and the lmm-CMA often outperforms NEWUOA and BFGS starting from about $2 \times D^2$ function evaluations with D being the search space dimension.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*global optimization, unconstrained optimization*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

Keywords

Benchmarking, Black-box optimization, expensive optimization, surrogate, CMA-ES

1. INTRODUCTION

The local-meta-model CMA-ES (lmm-CMA) introduced in [8] is an algorithm dedicated to optimization in contexts where the objective function is expensive to evaluate. The underlying algorithm is the well-known CMA-ES [7] where evaluations of the (expensive) objective function $f : \mathbb{R}^D \mapsto \mathbb{R}$ are replaced by approximated function values estimated by building for each point a quadratic meta-model $\hat{f}$ using the set of points already evaluated on f itself. The quality of the approximated solutions is appraised by tracking how the ranking of the solutions changes when a fraction of the candidate solutions is evaluated on f and new quadratic meta-models are built.

This paper evaluates a variant of the original lmm-CMA described in [8] on the BBOB 2013 expensive setup and compares it to CMA-ES without meta-modeling using the exact same maximal budget as well as to previously benchmarked algorithms namely the NEWUOA, BFGS and saACM algorithms. Section 2 describes in details the algorithm while Section 3 presents the different results obtained.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
Copyright 2013 ACM 978-1-4503-1964-5/13/07 ...$15.00.

2. THE lmm-CMA

The lmm-CMA algorithm builds on the $(\mu/\mu_w, \lambda)$-CMA-ES [6] and replaces the evaluation step of the λ candidate solutions by a surrogate-assisted procedure. More precisely, the algorithm is using a database $\mathcal{S}$ of solutions evaluated on the original objective function f stored as couples $(\mathbf{s}_i \in \mathbb{R}^D, y_i = f(\mathbf{s}_i))$. Until a minimum number of data points are stored in the database, the normal CMA-ES iterations are conducted. The minimal number of points required to start the surrogate-assisted procedure equals $\text{lsqDim} + 1 = D(D+3)/2 + 1 + 1$ that corresponds to the number of free parameters of one meta-model (see below) plus 1. Once this minimal number of points is in the database, the surrogate-assisted evaluation procedure (see Algorithm 1) is called to replace at each iteration the evaluations of the λ candidate solution on the objective function f. This procedure aims at providing an estimation of the ranking of the μ (out of λ) best candidate solutions to CMA-ES. Indeed, CMA-ES is a rank-based algorithm that only requires the ranking of the solutions (and not the exact function value of each solution) to perform all its updates.

The general idea of the surrogate-assisted procedure is to build for each candidate solution $\mathbf{x}_k$ a full quadratic meta-model $\hat{f}_{\mathbf{x}_k}$ which has $\text{lsqDim} = D(D+3)/2+1$ free parameters. This quadratic meta-model is then used to predict the function value at the corresponding solution $\mathbf{x}_k$ by evaluating it on $\hat{f}_{\mathbf{x}_k}$. The construction of the quadratic meta-model is described in detail in Section 2.1.

The quality of the ranking predicted by the construction of the meta-model for each candidate solution is ensured by evaluating a portion of the best individuals (parameters

$n_{\text{init}} \geq 0$ and $n_b > 0$ in Algorithm 1) on the original objective function f; performing anew the construction of the meta-models for each candidate solution and using the predicted ranking only if the ranking change is not too large. We refer to the pseudocode given in Algorithm 1 for the details and point out that it is different to the original code published in [8].

The differences have two origins: (i) we started from the code kindly provided by Stefan Kern and noticed differences between the code provided and the pseudocode published. (ii) In addition, we implemented more changes to improve the original algorithm and to deal with numerical instabilities.

First, we changed the acceptance criterion for the ranking. We remind that in the original algorithm, after a fraction of candidate solutions are evaluated on the original function, the surrogate-assisted procedure will be stopped if and only if the exact ranking of the μ best solutions stays the same. This acceptance criterion is more and more difficult to satisfy when the population size increases and leads to no speed-up w.r.t. the original CMA-ES [2]. We therefore compute the model-error between the old and new ranking as the sum of the rank differences (see line 15 of Algorithm 1) and accept the predicted ranking as soon as the model-error between the current and the previous ranking is not larger than a given quality-threshold that we have set after some parameter tuning on a few functions to $\lambda^2/20$.

Second, the original paper proposes to start the surrogate-assisted procedure after the first $2\times$lsqDim points are in the database and to use for each construction of the meta-model the closest $k_{\text{nn}} = 2 \times$ lsqDim points from the database $\mathcal{S}$. Instead, in order to save more function evaluations, we start to build the meta-models as soon as the database contains lsqDim$+1$ points and we use a number of points k_{nn} to build each meta-model that equals $\lceil \min(k_{\text{nn}}^{\text{target}}, \sqrt{(|\mathcal{S}| \times \text{lsqDim})}) \rceil$ where $|\mathcal{S}|$ is the number of points in the database when entering the surrogate-assisted evaluation procedure and $k_{\text{nn}}^{\text{target}} = 2\times$lsqDim (see also Algorithm 1). The model building is furthermore considered unsuccessful, if in a single coordinate both linear and quadratic coefficients are zero, in which case all candidate solutions of the iteration are evaluated on the original objective.

2.1 Construction of a meta-model

We describe in this section the construction of a full quadratic meta-model for a candidate solution called here query point. This construction uses the database $\mathcal{S}$ of points already visited and evaluated on the original objective function as well as a distance defined via the covariance matrix $\mathbf{C}$ and step-size σ of CMA-ES. This distance d is the Mahalanobis distance associated to the overall covariance matrix $\sigma^2\mathbf{C}$, i.e. for all $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{y} \in \mathbb{R}^D$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{C}^{-1}/\sigma^2(\mathbf{x}-\mathbf{y})} = \|\mathbf{M}(\mathbf{x}-\mathbf{y})\| \quad (1)$$

with $\|.\|$ the Euclidean norm and $\mathbf{M}$ in the RHS being equal to $\mathbf{M} = \frac{1}{\sigma}\mathbf{D}^{-\frac{1}{2}}\mathbf{B}^T$ with $\mathbf{B}$ an orthogonal matrix and $\mathbf{D}$ a diagonal matrix that stem from the eigen-decomposition of $\mathbf{C}$, i.e. $\mathbf{C} = \mathbf{B}\mathbf{D}\mathbf{B}^T$. It follows from the definition of $\mathbf{M}$ that $\mathbf{M}^T\mathbf{M} = \frac{1}{\sigma^2}\mathbf{C}^{-1}$.

Let $\mathbf{q}$ be a query point where a quadratic meta-model needs to be built. The expression of this meta-model with respect to the variable $\mathbf{z} = \mathbf{M}(\mathbf{x}-\mathbf{q})$ reads

$$\begin{aligned}\hat{f}_{\boldsymbol{\beta}}(\mathbf{x}) &= (\mathbf{x}-\mathbf{q})^T\mathbf{M}^T\mathbf{A}\mathbf{M}(\mathbf{x}-\mathbf{q}) + \mathbf{a}^T\mathbf{M}(\mathbf{x}-\mathbf{q}) + a_0 \quad (2)\\ &= \mathbf{z}^T\mathbf{A}\mathbf{z} + \mathbf{a}^T\mathbf{z} + a_0\end{aligned}$$

where $\boldsymbol{\beta} = [\mathbf{A}, \mathbf{a}, a_0]$ (with $\mathbf{A} \in \mathbb{R}^{D\times D}$ symmetric, $\mathbf{a} \in \mathbb{R}^D$ and $a_0 \in \mathbb{R}^+$) needs to be determined. To build the meta-model, we select from the database $\mathcal{S}$ the k_{nn} nearest points to $\mathbf{q}$ according to the Mahalanobis distance (1). Those k_{nn} nearest points are denoted $(\mathbf{s}_i, y_i = f(\mathbf{s}_i))_{1\leq i\leq k_{\text{nn}}}$ and assumed sorted ($(\mathbf{s}_{k_{\text{nn}}}, y_{k_{\text{nn}}})$ has the largest distance). We then determine the coefficients of $\boldsymbol{\beta}$ that minimize the weighted least square error

$$\mathcal{LS}(\boldsymbol{\beta}) = \sum_{i=1}^{k_{\text{nn}}} \mathrm{K}\left(\frac{d(\mathbf{s}_i, \mathbf{q})}{d(\mathbf{s}_{k_{\text{nn}}}, \mathbf{q})}\right)(\hat{f}_{\boldsymbol{\beta}}(\mathbf{s}_i) - y_i)^2 \ , \quad (3)$$

where we use for the kernel function $\mathrm{K}(\xi) = (1-\xi^2)^2$ and d is the Mahalanobis distance. The implementation of the solution of the least-square problem (3) uses the fact that the thought quadratic model is linear in the coefficients of $\boldsymbol{\beta}$, more precisely $\hat{f}_{\boldsymbol{\beta}}(\mathbf{s}_i) = \tilde{\mathbf{z}}_i.\tilde{\boldsymbol{\beta}}$ where $\mathbf{z}_i = \mathbf{M}(\mathbf{s}_i - \mathbf{q})$ and given a vector $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_D)^T$ the vector $\tilde{\mathbf{z}} \in \mathbb{R}^{D(D+3)/2+1}$ equals

$$\tilde{\mathbf{z}} = (\mathbf{z}_1^2, \ldots, \mathbf{z}_D^2, \mathbf{z}_1\mathbf{z}_2, \ldots, \mathbf{z}_{D-1}\mathbf{z}_D, \mathbf{z}_1, \ldots, \mathbf{z}_D, 1)$$

and $\tilde{\boldsymbol{\beta}} = (\mathbf{A}_{11}, \ldots, \mathbf{A}_{DD}, 2\mathbf{A}_{12}, \ldots, 2\mathbf{A}_{D-1,D}, \mathbf{a}_1, \ldots, \mathbf{a}_D, a_0)^T$. Defining the matrix $\mathbf{W} = \mathrm{diag}\ \sqrt{\mathrm{K}\left(d(\mathbf{s}_i, \mathbf{q})/d(\mathbf{s}_{k_{\text{nn}}}, \mathbf{q})\right)}$, the weighted Least Square error in (3) writes

$$\mathcal{LS}(\tilde{\boldsymbol{\beta}}) = \|\mathbf{W}\mathbf{Z}\tilde{\boldsymbol{\beta}} - \mathbf{W}\mathbf{Y}\|^2$$

where

$$\mathbf{Z} = \begin{pmatrix} \tilde{\mathbf{z}}_1 \\ \tilde{\mathbf{z}}_2 \\ \cdots \\ \tilde{\mathbf{z}}_{k_{\text{nn}}} \end{pmatrix}, \mathbf{Y} = (y_1, \ldots, y_{k_{\text{nn}}})^T$$

and is solved in our Matlab implementation using the back-slash operator, i.e.

$$\tilde{\boldsymbol{\beta}} = \mathbf{W}\mathbf{Z}\tilde{\boldsymbol{\beta}} \backslash \mathbf{W}\mathbf{Y} \ .$$

The estimate of the function value at the query point $\mathbf{q}$ corresponds to the last coefficient of $\tilde{\boldsymbol{\beta}}$ encoding the constant term of the quadratic model in the chosen representation (2).

2.2 Parameter tuning

The setting for the parameters used for the surrogate assisted procedure is indicated in the pseudo-code in Figure 1. One specific parameter was tuned using the BBOB benchmark suite, namely the quality_threshold parameter: two experiments were conducted, one with quality_threshold equal to 1 and another one with a very large value equal to 10^{12}. Then the quality_threshold parameter was tuned using a few trials on single functions to a value small enough so that the functions not solved with the threshold value of 10^{12} but solved with a threshold equal to 1 could be solved, and large enough to not loose much on the other functions. In addition, the default parameters of CMA-ES were used.

3. RESULTS

The following subsections present the results after running the Matlab lmm-CMA code on the BBOB'2013 testbed with

Algorithm 1 Pseudocode of the ranking prediction in the $(\mu/\mu_w, \lambda)$-lmm-CMA variant benchmarked here.

Require: λ solutions $\mathbf{x}_k$ $(1 \le k \le \lambda)$ and at least $\mathrm{lsqDim} + 1$ points $\mathbf{s}_i$ with true function value $y_i = f(\mathbf{s}_i)$ in database $\mathcal{S}$

Parameters: $n_{\mathrm{init}} \ge 0$, $n_b > 0$, a quality_threshold ≥ 0, and the number of nearest neighbors k_{nn} used in the meta modeling (here n_{init} is initially chosen as 1 in each CMA-ES run and later adapted in lines 16–17, $n_b = \max\{\lceil \lambda/20 \rceil, 1\}$, quality_threshold $= \lambda^2/20$, and $k_{\mathrm{nn}} = \lceil \min\{k_{\mathrm{nn}}^{\mathrm{target}}, \sqrt{|\mathcal{S}| \times \mathrm{lsqDim}}\} \rceil$ with $k_{\mathrm{nn}}^{\mathrm{target}} = 2 \times \mathrm{lsqDim}$)

Ensure: returns a ranking of the best μ solutions

1: **for** all $\mathbf{x}_k$ $(1 \le k \le \lambda)$ **do**
2: *model building:* build local model $\hat{f}_{\boldsymbol{\beta}_k}$ based on k_{nn} individuals in database $\mathcal{S}$ closest to $\mathbf{x}_k$
3: *rank:* generate $\mathrm{ranking}_0^\mu$ of the μ best individuals based on the function values $\hat{f}_{\boldsymbol{\beta}_k}(\mathbf{x}_k)$
4: evaluate the n_{init} best individuals (based on $\mathrm{ranking}_0^\mu$) on true function f and add them with their function values to $\mathcal{S}$
5: **for** all $\mathbf{x}_k$ $(1 \le k \le \lambda)$ **do**
6: *model building:* build local model $\hat{f}_{\boldsymbol{\beta}_k}$ based on k_{nn} individuals in database $\mathcal{S}$ closest to $\mathbf{x}_k$
7: *rank:* update $\mathrm{ranking}_0^\mu$ of the μ best individuals based on the function values $\hat{f}_{\boldsymbol{\beta}_k}(\mathbf{x}_k)$
8: set modelerror $= +\inf$; set counter $i = 0$
9: **while** not all λ individuals evaluated on true objective function **a**nd modelerror $>$ quality_threshold **do**
10: set $i = i + 1$
11: evaluate the next n_b best unevaluated individuals (based on previous $\mathrm{ranking}_{i-1}^\mu$) and add them with their function values to $\mathcal{S}$
12: **for** all $\mathbf{x}_k$ $(1 \le k \le \lambda)$ **do**
13: *model building:* build local model $\hat{f}_{\boldsymbol{\beta}_k}$ based on k_{nn} individuals in database $\mathcal{S}$ closest to $\mathbf{x}_k$
14: *rank:* generate $\mathrm{ranking}_i^\mu$ of the μ best individuals based on the function values $\hat{f}_{\boldsymbol{\beta}_k}(\mathbf{x}_k)$
15: update modelerror between old and new ranking as $\sum_{1 \le j \le \mu} |\mathrm{ranking}_{i-1}^\mu(j) - \mathrm{ranking}_i^\mu(j)|$
16: **if** $i > 2$ **then** $n_{\mathrm{init}} = \min(\lambda, n_{\mathrm{init}} + n_b)$
17: **if** $i < 2$ **then** $n_{\mathrm{init}} = \max(0, n_{\mathrm{init}} - n_b)$
18: **return** $\mathrm{ranking}_i^\mu$

a maximum budget of $400(D+2)$ function evaluations. The lmm-CMA was run with independent restarts—doubling its population size after each restart from an initial $4 + \lfloor 3 \log D \rfloor$ (IPOP-CMA-ES setting, [1]). The initial mean vector of the search distribution was set to 0^D and the initial step size to 2. The `EvalInitialX` being on, the initial mean vector was evaluated. Other parameters were set according to the standard CMA-ES recommendations and we refer to the source code which is available at `http://canadafrance.gforge.inria.fr/lmmcmaes/` for details.

3.1 The lmm-CMA in the Expensive BBOB'2013 Setting

Figures 1 and 2 and Table 1 present the results of the lmm-CMA from experiments according to [4] on the benchmark functions given in [3, 5] in the expensive scenario.

Three main observations can be made with respect to the lmm-CMA and the expensive scenario: Firstly, the lmm-CMA can solve all BBOB'2013 functions except for the multi-modal f_{19} (and in 20-D, also not f_{23}) up to the run-length based target values just not reached by the artificial GECCO-BBOB-2009 best algorithm for a budget of $50 \times D$ evaluations; however, not all 15 runs are successful in all dimensions due to the restricted run length of $400(D+2)$ function evaluations. Secondly, the median ERT is only a factor of about 2–3 worse than for the artificial GECCO-BBOB-2009 best algorithm for all investigated run lengths in 5-D and 20-D (Fig.1). The 90%-ile of the ERT is a factor of ≤ 25 larger than the ERT of the 2009 artificial best algorithm. Last, the ERT of the lmm-CMA is lower than the one of the GECCO-BBOB-2009 best algorithm on f_7, f_{17}, and f_{18} in 5-D and 20-D (Table 1) whereas the results are significant only for a few run-length based targets (low budget for f_{18} in 20-D, $3 \times D$ on f_{17} in 20-D, $10 \times D$ for f_{17} in 5-D, and for $50 \times D$ for f_7 in 20-D and f_{18} in 5-D).

3.2 Comparison with IPOP-CMA-ES without Meta-Modeling

To compare the lmm-CMA with its counterpart without meta-modeling, the IPOP-CMA-ES, we run the IPOP-CMA-ES for the same maximal number of $400(D+2)$ function evaluations and the same setting than for the lmm-CMA (denoted IPOP400D). Due to space limitations, we cannot show all results of the pairwise comparison between the two algorithms, but refer to Fig. 3 and Table 2 for the main results. We observe that, on the one hand, the lmm-CMA shows improved performances on f_1, f_2, f_7, f_9, f_{10}, f_{11}, f_{12}, f_{13}, and f_{14} in most of the dimensions. There are eight functions overall in 20-D that can be solved up to a target level of 10^{-8} (within the $400(D+2)$ budget) by the lmm-CMA but not by the IPOP400D without meta-modeling. As both algorithms perform the evaluations on the true objective function until the database is large enough for the model building, this increase in performance is achieved in the later stages of the optimization. The plots in Fig. 3 show the equivalence of the two algorithms in the early stages and the improvement gained by the meta modeling nicely for all of the function subgroups. Though the time, when the ECDF plots of the two algorithms split in 20-D, depends on the function class, it lies in all cases between $10 \times D$ and $50 \times D$ function evaluations—note that the meta-model is not learned within the first $D(D+3)/2 + 2$ function evaluations, which equals about $11 \times D$ in 20-D.

When compared to the earlier benchmarked IPOP-CMA-ES with a maximal number of function evaluations of $10^6 D$ [12], the lmm-CMA is significantly faster by a factor of about

2 on f_8 and f_9, by a factor of about 3 on f_2 and f_{10}, by a factor of about 4 on f_{11} and f_{14}, and by a factor of 5.8 on the sphere to reach the target value of 10^{-7} in 20-D (results not shown due to space restrictions). The improvement factors over the IPOP-CMA-ES in 5-D are in a comparable range.

Though most of the functions show an improvement of the lmm-CMA over the IPOP-CMA-ES without meta-modeling, there seems to be one drawback of using the meta-model: the impact on the attractive sector function f_6 is significantly detrimental. Here, the meta-modeling slows down the optimization by a factor of about 50 for target precision $\Delta f_{\text{opt}} = 1$ and does not reach smaller target values whereas the original IPOP-CMA-ES reaches target values of about 10^{-7} in the budget of $400 \times D$ function evaluations. Other performance decreases, however, can not be observed.

3.3 Comparison with Other Optimizers

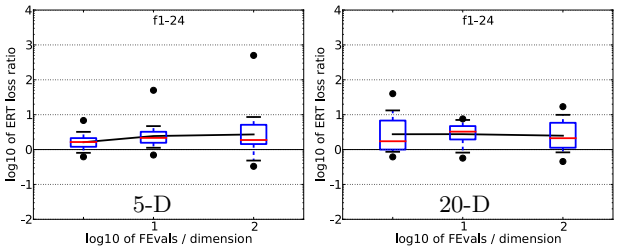
Finally, we compare the lmm-CMA and its version without meta-modeling with other algorithms that have been reported to have good results in the expensive scenario of BBOB. To this end, we postprocessed the online available data sets (see e.g. `http://coco.lri.fr/BBOB2009/rawdata/`) of the BFGS [10], NEWUOA [11], and IPOP-sa-ACM [9] algorithms. Figure 3 shows the ECDF plots for 20-D in the expensive scenario and Table 2 presents the ERT ratios of all algorithms for the standard (fixed) BBOB targets.

The ECDF graphs of Fig. 3 show thereby a quite similar performance between the lmm-CMA and the also meta-model assisted IPOP-sa-ACM in all function classes. Only for very short run lengths up to $10 \times D$, lmm-CMA is faster than IPOP-sa-ACM, most likely because the initial point is the middle of the search domain (all zeros). Except for the multi-modal functions f_{15}–f_{19}, where NEWUOA is always worse, the lmm-CMA is outperformed by NEWUOA in the early optimization stages (up to about $30 \times D$ function evaluations) and for the moderate and weakly-structured multi-modal functions where NEWUOA is better for all budgets until $1000 \times D$. The IPOP-sa-ACM is furthermore better than the lmm-CMA in the later stages on the moderate functions and BFGS is better than the lmm-CMA on the separable and ill-conditioned functions in the beginning and the middle stages of the optimization respectively.

The largest performance gap to the GECCO-BBOB-2009 best algorithm can be observed for the separable problems while over all functions, the lmm-CMA is at most a factor of about 5 worse than the artificial best algorithm of GECCO-BBOB-2009 (within the run-length based target values of the expensive scenario). For the fixed target scenario of Table 2, it furthermore becomes obvious that the other algorithms have been run in part much longer than the $400 \times D$ function evaluations of the lmm-CMA and the IPOP-CMA-ES without meta-modeling. However, the lmm-CMA allows to solve seven of the 24 functions for all 15 instances up to a precision of 10^{-8} and on additional five functions, the lmm-CMA reaches an accuracy of 10^{-7} for at least one run. The IPOP-CMA-ES version without meta-modeling, on the other hand, reaches a target value of 10^{-7} in the given budget of $400 \times D$ function evaluations only on four functions in 20-D.

3.4 CPU Timing

In order to evaluate the CPU timing of the algorithm, we have run the Matlab lmm-CMA code on the function f_8 with



	f_1–f_{24} in 5-D, maxFE/D=578					
#FEs/D	best	10%	25%	**med**	75%	90%
2	0.62	0.79	1.1	1.6	2.2	3.4
10	0.70	0.97	1.6	2.2	3.3	5.0
100	0.33	0.45	1.4	1.9	5.2	9.7
1e3	0.31	0.56	0.83	1.8	5.8	21
RL_{US}/D	6e2	6e2	6e2	6e2	6e2	6e2
	f_1–f_{24} in 20-D, maxFE/D=442					
#FEs/D	best	10%	25%	**med**	75%	90%
2	0.62	0.83	1.0	1.7	6.9	16
10	0.57	0.73	1.7	3.3	4.7	7.1
100	0.45	0.73	1.1	2.1	5.9	11
1e3	0.22	0.32	1.0	2.5	7.6	25
RL_{US}/D	4e2	4e2	4e2	4e2	4e2	4e2

Figure 1: ERT **loss ratio versus the budget for lmm-CMA (both in number of f-evaluations divided by dimension). The target value f_t for a given budget** FEvals **is the best target f-value reached within the budget by the given algorithm. Shown is the** ERT **of the given algorithm divided by best** ERT **seen in GECCO-BBOB-2009 for the target f_t, or, if the best algorithm reached a better target within the budget, the budget divided by the best** ERT**. Line: geometric mean. Box-Whisker error bar: 25-75%-ile with median (box), 10-90%-ile (caps), and minimum and maximum** ERT **loss ratio (points). The vertical line gives the maximal number of function evaluations in a single trial in this function subset.**

restarts for at least 30 seconds and until a maximum budget equal to $400(D+2)$ is reached. The code was run on a Mac Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz with 1 processor and 4 cores. However, not all the full CPU available is exploited by Matlab through parallelization, not more than 200% (out of 400 %) of the CPU was used. The time per function evaluation for dimensions 2, 3, 5, 10, 20, 40 equals 3.8, 4.6, 7.8, 20, 170, and 4400 milliseconds respectively. Restarts happened only up to 5-D.

Acknowledgements

This work was supported by the grants ANR-2010-COSI-002 (SIMINOLE) and the ANR-12-MONU-0009 (NumBBO) of the French National Research Agency.

4. REFERENCES

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. pages 1769–1776. IEEE Press, 2005.

[2] Z. Bouzarkouna, A. Auger, and D. Y. Ding. Investigating the local-meta-model CMA-ES for large population sizes. In *3rd European event on*

5-D

#FEs/D	0.5	1.2	3	10	50	#succ
$\mathbf{f_1}$	*2.5e+1*:4.8	*1.6e+1*:7.6	*1.0e-8*:12	*1.0e-8*:12	*1.0e-8*:12	15/15
	1.2(2)	1.5(1)	9.1(0.6)	9.1(0.6)	9.1(0.6)	15/15
$\mathbf{f_2}$	*1.6e+6*:2.9	*4.0e+5*:11	*4.0e+4*:15	*6.3e+2*:58	*1.0e-8*:95	15/15
	1.6(2)	0.83(0.9)	2.4(2)	2.5(0.7)	5.5(1)	15/15
$\mathbf{f_3}$	*1.6e+2*:4.1	*1.0e+2*:15	*6.3e+1*:23	*2.5e+1*:73	*1.0e+1*:716	15/15
	1.4(1)	0.92(0.8)	1.4(1)	2.4(1)	0.45(0.1)	15/15
$\mathbf{f_4}$	*2.5e+2*:2.6	*1.6e+2*:10	*1.0e+2*:19	*4.0e+1*:65	*1.6e+1*:434	15/15
	0.51(0.4)	0.79(1)	0.93(1)	2.2(0.9)	1.3(1)	15/15
$\mathbf{f_5}$	*6.3e+1*:4.0	*4.0e+1*:10	*1.0e-8*:10	*1.0e-8*:10	*1.0e-8*:10	15/15
	3.1(2)	1.9(0.8)	5.0(1)	5.0(1)	5.0(1)	15/15
$\mathbf{f_6}$	*1.0e+5*:3.0	*2.5e+4*:8.4	*1.0e+2*:16	*2.5e+1*:54	*2.5e-1*:254	15/15
	1.6(2)	1.4(1)	4.7(5)	3.8(4)	4.5(3)	15/15
$\mathbf{f_7}$	*1.6e+2*:4.2	*1.0e+2*:6.2	*2.5e+1*:20	*4.0e+0*:54	*1.0e+0*:324	15/15
	1.2(1)	1.3(1)	1.5(1)	2.3(4)	0.92(1.0)	15/15
$\mathbf{f_8}$	*1.0e+4*:4.6	*6.3e+3*:6.8	*1.0e+3*:18	*6.3e+1*:54	*1.6e+0*:258	15/15
	1.0(1)	0.96(1)	1.6(1)	1.5(0.6)	1.7(2)	15/15
$\mathbf{f_9}$	*2.5e+1*:20	*1.6e+1*:26	*1.0e+1*:35	*4.0e+0*:62	*1.6e-2*:256	15/15
	3.7(2)	3.3(1)	2.7(0.9)	2.7(2)	2.4(1)	15/15
$\mathbf{f_{10}}$	*2.5e+6*:2.9	*6.3e+5*:7.0	*2.5e+5*:17	*6.3e+3*:54	*2.5e+1*:297	15/15
	2.3(3)	1.5(1)	1.1(0.9)	1.5(0.4)	0.83(0.4)	15/15
$\mathbf{f_{11}}$	*1.0e+6*:3.0	*6.3e+4*:6.2	*6.3e+2*:16	*6.3e+1*:74	*6.3e-1*:298	15/15
	1.4(2)	2.1(2)	2.6(1.0)	1.9(1)	1.3(0.3)	15/15
$\mathbf{f_{12}}$	*4.0e+7*:3.6	*1.6e+7*:7.6	*4.0e+6*:19	*1.6e+4*:52	*1.0e+0*:268	15/15
	0.80(0.6)	1.6(2)	1.6(0.8)	1.7(0.6)	1.4(1)	15/15
$\mathbf{f_{13}}$	*1.0e+3*:2.8	*6.3e+2*:8.4	*4.0e+2*:17	*6.3e+1*:52	*6.3e-2*:264	15/15
	1.4(2)	1.6(2)	1.7(1.0)	1.8(0.5)	1.6(0.4)	15/15
$\mathbf{f_{14}}$	*1.6e+1*:3.0	*1.0e+1*:10	*6.3e+0*:15	*2.5e-1*:53	*1.0e-5*:251	15/15
	1.1(1)	0.62(0.6)	0.81(0.7)	1.6(0.5)	1.8(0.2)	15/15
$\mathbf{f_{15}}$	*1.6e+2*:3.0	*1.0e+2*:13	*6.3e+1*:24	*4.0e+1*:55	*1.6e+1*:289	5/5
	1.3(2)	1.3(1)	1.5(0.9)	1.6(1)	1.4(2)	15/15
$\mathbf{f_{16}}$	*4.0e+1*:4.8	*2.5e+1*:16	*1.6e+1*:46	*1.0e+1*:120	*4.0e+0*:334	15/15
	1.7(2)	1.8(2)	2.7(3)	2.0(2)	1.3(0.8)	15/15
$\mathbf{f_{17}}$	*1.0e+1*:5.2	*6.3e+0*:26	*4.0e+0*:57	*2.5e+0*:110	*6.3e-1*:412	15/15
	1.7(2)	0.91(0.8)	0.70(0.5)	**0.55**(0.2)$^{\downarrow}$	0.62(0.3)	15/15
$\mathbf{f_{18}}$	*6.3e+1*:3.4	*4.0e+1*:7.2	*2.5e+1*:20	*1.6e+1*:58	*1.6e+0*:318	15/15
	1.3(1)	1.6(2)	0.92(0.8)	0.73(0.4)	**0.52**(0.2)$^{\downarrow 4}$	15/15
$\mathbf{f_{19}}$	*1.6e-1*:172	*1.0e-1*:242	*6.3e-2*:675	*4.0e-2*:3078	*2.5e-2*:4946	15/15
	55(57)	56(64)	30(33)	14(15)	*∞2800*	0/15
$\mathbf{f_{20}}$	*6.3e+3*:5.1	*4.0e+3*:8.4	*4.0e+1*:15	*2.5e+0*:69	*1.0e+0*:851	15/15
	1.2(0.9)	0.83(0.5)	1.8(1)	4.0(6)	15(17)	3/15
$\mathbf{f_{21}}$	*4.0e+1*:3.9	*2.5e+1*:11	*1.6e+1*:31	*6.3e+0*:73	*1.6e+0*:347	5/5
	1.9(2)	1.4(2)	0.93(0.7)	1.6(0.8)	2.7(4)	13/15
$\mathbf{f_{22}}$	*6.3e+1*:3.6	*4.0e+1*:15	*2.5e+1*:32	*1.0e+1*:71	*1.6e+0*:341	5/5
	1.6(2)	1.4(1)	1.3(1)	2.5(3)	4.2(5)	12/15
$\mathbf{f_{23}}$	*1.0e+1*:3.0	*6.3e+0*:9.0	*4.0e+0*:33	*2.5e+0*:84	*1.0e+0*:518	15/15
	1.9(2)	2.6(2)	2.6(3)	7.1(6)	10(11)	6/15
$\mathbf{f_{24}}$	*6.3e+1*:15	*4.0e+1*:37	*4.0e+1*:37	*2.5e+1*:118	*1.6e+1*:692	15/15
	0.77(1.0)	2.0(2)	2.0(2)	1.7(1)	0.91(0.6)	15/15

20-D

#FEs/D	0.5	1.2	3	10	50	#succ
$\mathbf{f_1}$	*6.3e+1*:24	*4.0e+1*:42	*1.0e-8*:43	*1.0e-8*:43	*1.0e-8*:43	15/15
	2.5(1)	2.5(1)	10(0.2)	10(0.2)	10(0.2)	15/15
$\mathbf{f_2}$	*4.0e+6*:29	*2.5e+6*:42	*1.0e+5*:65	*1.0e+4*:207	*1.0e-8*:412	15/15
	0.53(0.5)	0.68(0.7)	7.2(2)	3.8(0.8)	14(1)	15/15
$\mathbf{f_3}$	*6.3e+2*:33	*4.0e+2*:44	*1.6e+2*:109	*1.0e+2*:255	*2.5e+1*:3277	15/15
	1.0(1)	2.3(1)	5.9(2)	4.8(0.9)	2.1(1.0)	14/15
$\mathbf{f_4}$	*6.3e+2*:22	*4.0e+2*:91	*2.5e+2*:250	*1.6e+2*:332	*6.3e+1*:1927	15/15
	1.3(2)	2.2(0.9)	2.0(0.6)	3.0(0.8)	2.0(2)	13/15
$\mathbf{f_5}$	*2.5e+2*:19	*1.6e+2*:34	*1.0e-8*:41	*1.0e-8*:41	*1.0e-8*:41	15/15
	1.8(0.5)	2.1(0.6)	6.1(1)	6.1(1)	6.1(1)	15/15
$\mathbf{f_6}$	*2.5e+5*:16	*6.3e+4*:43	*1.6e+4*:62	*1.6e+2*:353	*1.6e+1*:1078	15/15
	1.6(1)	1.6(1)	2.0(0.9)	6.8(3)	9.0(6)	11/15
$\mathbf{f_7}$	*1.0e+3*:11	*4.0e+2*:39	*2.5e+2*:74	*6.3e+1*:319	*1.0e+1*:1351	15/15
	0.49(0.7)	1.2(0.9)	1.7(1)	1.0(0.2)	**0.48**(0.1)$^{\downarrow 4}$	15/15
$\mathbf{f_8}$	*4.0e+4*:19	*2.5e+4*:35	*4.0e+3*:67	*2.5e+2*:231	*1.6e+1*:1470	15/15
	2.2(2)	2.5(1)	3.3(1.0)	2.6(0.5)	1.3(0.5)	15/15
$\mathbf{f_9}$	*1.0e+2*:357	*6.3e+1*:560	*4.0e+1*:684	*2.5e+1*:756	*1.0e+1*:1716	15/15
	1.6(0.4)	1.3(0.2)	1.1(0.2)	1.2(0.2)	2.1(0.5)	15/15
$\mathbf{f_{10}}$	*1.6e+6*:15	*1.0e+6*:27	*4.0e+5*:70	*6.3e+4*:231	*4.0e+3*:1015	15/15
	3.7(3)	3.5(3)	4.2(1)	2.3(0.5)	1.0(0.2)	15/15
$\mathbf{f_{11}}$	*4.0e+4*:11	*2.5e+3*:27	*1.6e+2*:313	*1.0e+2*:481	*1.0e+1*:1002	15/15
	1.3(1)	1.4(1.0)	4.5(3)	3.7(0.6)	2.1(0.2)	15/15
$\mathbf{f_{12}}$	*1.0e+8*:23	*6.3e+7*:39	*2.5e+7*:76	*4.0e+6*:209	*1.0e+1*:1042	15/15
	2.0(2)	2.4(1)	2.6(0.7)	1.9(0.3)	1.1(0.1)	15/15
$\mathbf{f_{13}}$	*1.6e+3*:28	*1.0e+3*:64	*6.3e+2*:79	*4.0e+1*:211	*2.5e+0*:1724	15/15
	1.6(0.7)	2.6(1)	3.3(0.9)	4.2(0.4)	1.1(0.2)	15/15
$\mathbf{f_{14}}$	*2.5e+1*:15	*1.6e+1*:42	*1.0e+1*:75	*1.6e+0*:219	*6.3e-4*:1106	15/15
	3.8(5)	2.9(2)	3.0(1)	2.2(0.5)	1.9(0.1)	15/15
$\mathbf{f_{15}}$	*6.3e+2*:15	*4.0e+2*:67	*2.5e+2*:292	*1.6e+2*:846	*1.0e+2*:1671	15/15
	1.4(2)	1.3(0.9)	0.77(0.4)	0.72(0.3)	0.86(0.3)	15/15
$\mathbf{f_{16}}$	*4.0e+1*:26	*2.5e+1*:127	*1.6e+1*:540	*1.6e+1*:540	*1.0e+1*:1384	15/15
	4.8(7)	8.1(6)	2.7(0.9)	2.7(0.9)	1.2(0.4)	15/15
$\mathbf{f_{17}}$	*1.6e+1*:11	*1.0e+1*:63	*6.3e+0*:305	*4.0e+0*:468	*1.0e+0*:1030	15/15
	0.62(0.8)	1(1)	**0.65**(0.4)$^{\downarrow}$	0.79(0.3)	1.4(0.6)	14/15
$\mathbf{f_{18}}$	*4.0e+1*:116	*2.5e+1*:252	*1.6e+1*:430	*1.0e+1*:621	*4.0e+0*:1090	15/15
	0.41(0.6)$^{\downarrow}$	**0.57**(0.3)$^{\downarrow 2}$	0.77(0.3)	0.76(0.3)	0.85(0.3)	15/15
$\mathbf{f_{19}}$	*1.6e-1*:2.5e5	*1.0e-1*:3.4e5	*6.3e-2*:3.4e5	*4.0e-2*:3.4e5	*2.5e-2*:3.4e5	3/15
	∞	∞	∞	∞	*∞8800*	0/15
$\mathbf{f_{20}}$	*1.6e+4*:38	*1.0e+4*:42	*2.5e+2*:62	*2.5e+0*:250	*1.6e+0*:2536	15/15
	0.91(0.9)	1.4(2)	3.9(0.9)	6.6(1)	23(28)	2/15
$\mathbf{f_{21}}$	*6.3e+1*:36	*4.0e+1*:77	*4.0e+1*:77	*1.6e+1*:456	*4.0e+0*:1094	15/15
	3.5(1)	2.5(1.0)	2.5(1.0)	1.3(2)	4.3(5)	11/15
$\mathbf{f_{22}}$	*6.3e+1*:45	*4.0e+1*:68	*4.0e+1*:68	*1.6e+1*:231	*6.3e+0*:1219	15/15
	3.1(1)	5.8(2)	5.8(2)	5.9(6)	3.8(5)	11/15
$\mathbf{f_{23}}$	*6.3e+0*:29	*4.0e+0*:118	*2.5e+0*:306	*2.5e+0*:306	*1.0e+0*:1614	15/15
	1.9(3)	8.2(9)	408(462)	408(462)	*∞8800*	0/15
$\mathbf{f_{24}}$	*2.5e+2*:208	*1.6e+2*:918	*1.0e+2*:6628	*6.3e+1*:9885	*4.0e+1*:31629	15/15
	0.74(0.2)$^{\downarrow}$	1.1(0.8)	1.4(1)	1.2(1)	1.2(1)	3/15

Table 1: Expected running time (ERT in number of function evaluations) of the lmm-CMA divided by the best ERT measured during BBOB-2009. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear in the second row of each cell, the best ERT (preceded by the target Δf-value in *italics*) in the first. #succ is the number of trials that reached the target value of the last column. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Bold entries are statistically significantly better (according to the rank-sum test) compared to the best algorithm in BBOB-2009, with $p = 0.05$ or $p = 10^{-k}$ when the number $k > 1$ is following the $\downarrow$ symbol, with Bonferroni correction by the number of functions.

Bio-inspired algorithms for continuous parameter optimisation (EvoNUM'10). Springer, 2010.

[3] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009. Updated February 2010.

[4] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[6] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. pages 282–291. Springer, 2004.

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

[8] S. Kern, N. Hansen, and P. Koumoutsakos. Local meta-models for optimization using evolution strategies. In *Parallel Problem Solving from Nature - PPSN IX*, pages 939–948. Springer, 2006.

[9] I. Loshchilov, M. Schoenauer, and M. Sebag. Black-box optimization benchmarking of ipop-saacm-es and bipop-saacm-es on the bbob-2012 noiseless testbed. In T. Soule, editor, *GECCO 2012 (Companion)*. ACM, 2012.

[10] R. Ros. Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. In Rothlauf [13], pages 2409–2414.

[11] R. Ros. Benchmarking the NEWUOA on the BBOB-2009 function testbed. In Rothlauf [13], pages 2421–2428.

[12] R. Ros. Black-box optimization benchmarking the IPOP-CMA-ES on the noiseless testbed: comparison to the BIPOP-CMA-ES. In *GECCO 2010 (Companion)*, pages 1503–1510. ACM, 2010.

[13] F. Rothlauf, editor. *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, Companion Material*. ACM, 2009.

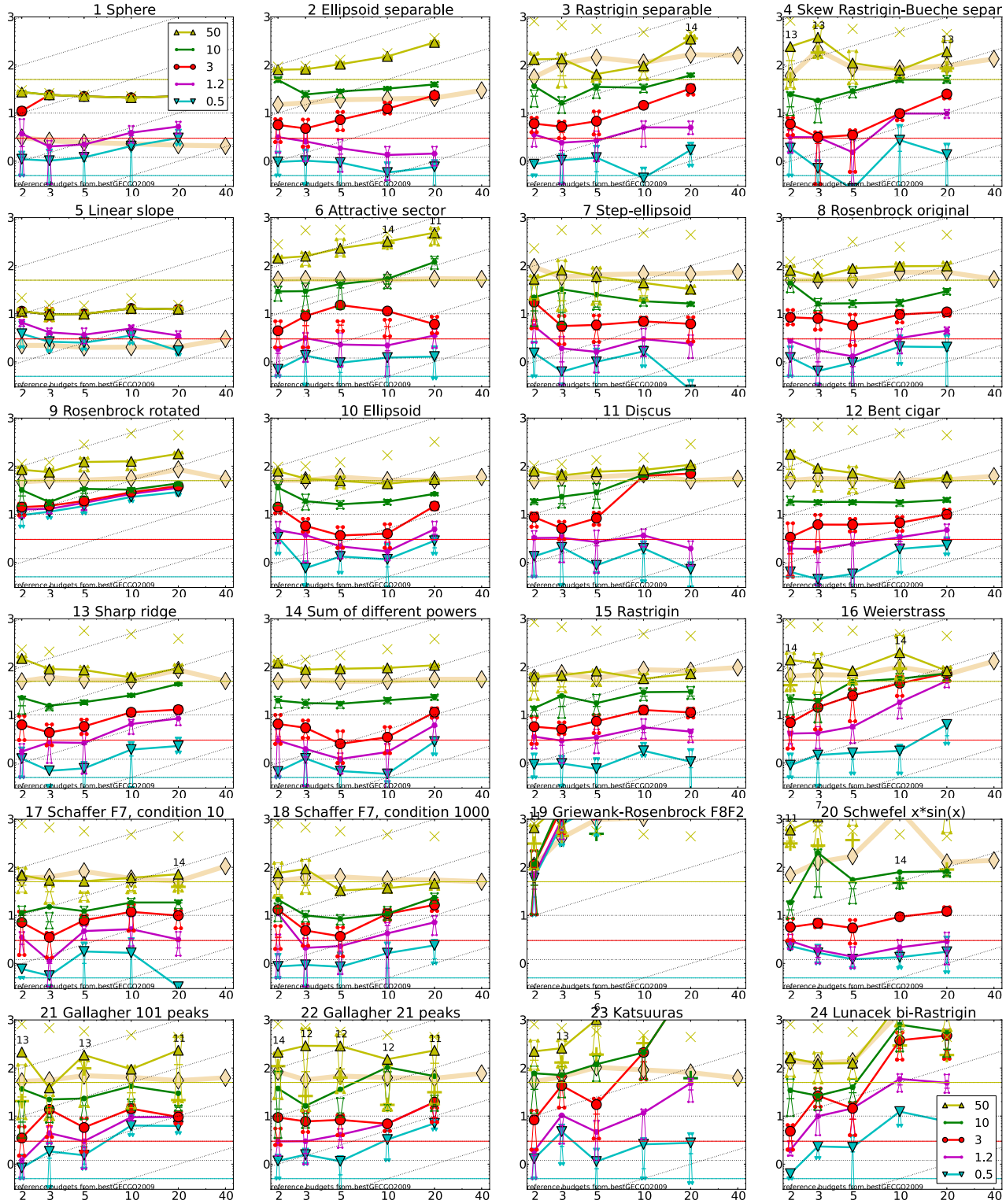


Figure 2: Expected number of f-evaluations (ERT, lines) to reach $f_{\mathrm{opt}} + \Delta f$; median number of f-evaluations ($+$) to reach the most difficult target that was reached not always but at least once; maximum number of f-evaluations in any trial ($\times$); interquartile range with median (notched boxes) of simulated runlengths to reach $f_{\mathrm{opt}} + \Delta f$; all values are divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown is the ERT for targets just not reached by the artificial GECCO-BBOB-2009 best algorithm within the given budget $k \times \mathrm{DIM}$, where k is shown in the legend. Numbers above ERT-symbols indicate the number of trials reaching the respective target. The light thick line with diamonds indicates the respective best result from BBOB-2009 for the most difficult target. Slanted grid lines indicate a scaling with $\mathcal{O}(\mathrm{DIM})$ compared to $\mathcal{O}(1)$ when using the respective 2009 best algorithm.

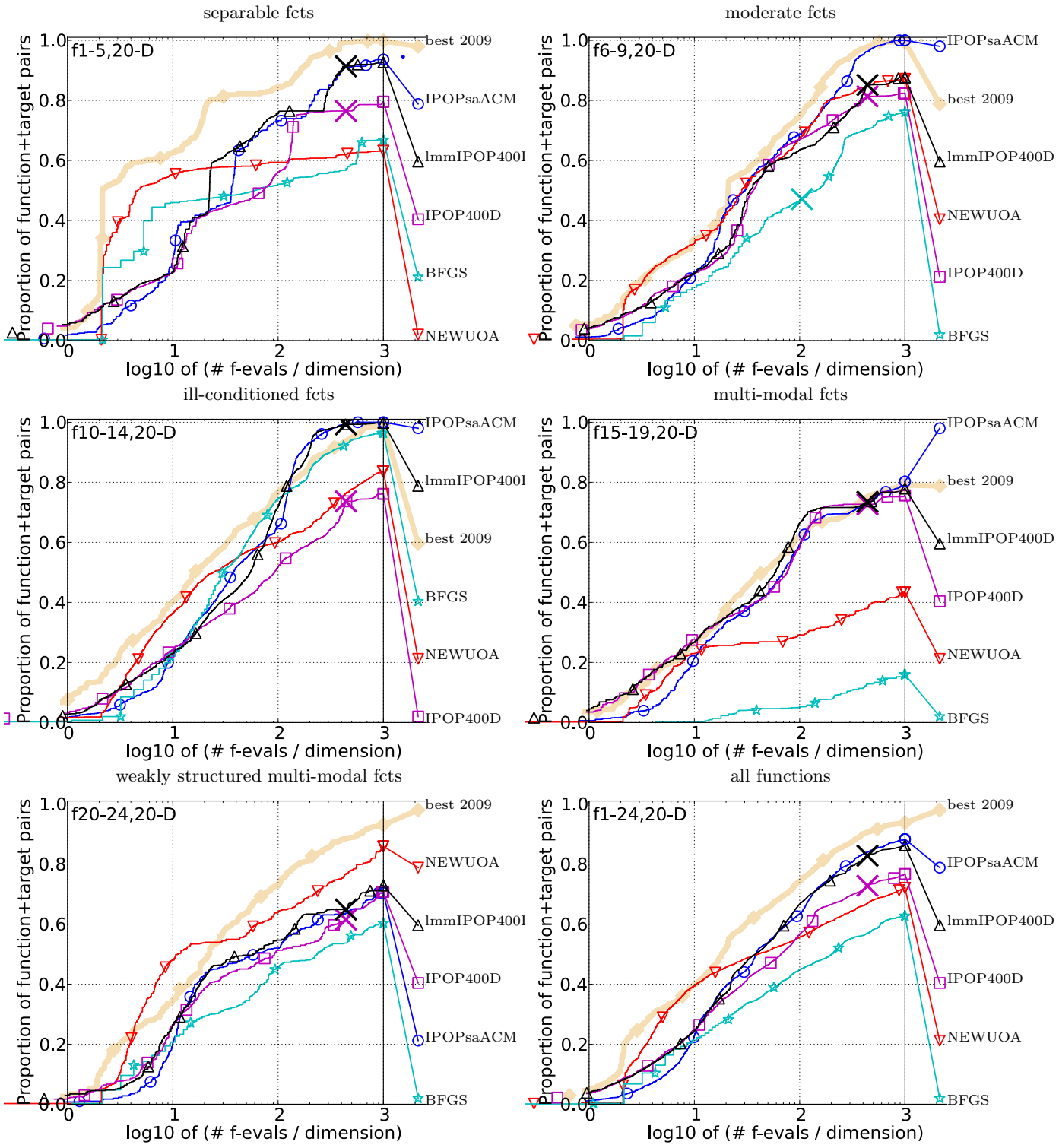


Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 31 target f-values just not reached by the artificial GECCO-BBOB-2009 best algorithm within 31 reference budgets in $[0.5 \ldots 50] \times \mathrm{DIM}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the artificial GECCO-BBOB-2009 best algorithm, i.e., the best ERT observed during BBOB 2009 for each single target.

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	43	43	43	43	43	43	15/15
IPOPsaACM	3.9(0.2)	5.0(0.4)	6.5(0.5)	9.5(0.7)	13(0.7)	16(0.8)	15/15
NEWUOA	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	1.0(0.0)	15/15
BFGS	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	**1**(0)	15/15
IPOP400D	6.6(0.8)	13(1)	19(1)	31(2)	44(2)	56(3)	15/15
lmmIPOP400D	5.6(0.2)	6.3(0.1)	6.8(0.1)	7.8(0.1)	8.9(0.2)	10(0.2)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	385	386	387	390	391	393	15/15
IPOPsaACM	**7.3**(1)	**8.3**(2)$^{\star 2}$	**8.9**(2)$^{\star 3}$	**10**(2)$^{\star 4}$	**10**(1)$^{\star 4}$	**10**(1)$^{\star 4}$	15/15
NEWUOA	18(8)	42(21)	71(36)	125(43)	174(51)	219(67)	15/15
BFGS	20(4)	24(5)	26(4)	27(3)	28(3)	28(3)	15/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	8.4(1)	11(2)	12(2)	13(2)	14(2)	15(1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	5066	7626	7635	7643	7646	7651	15/15
IPOPsaACM	**11**(15)	∞	∞	∞	∞	∞ *2e7*	0/5
NEWUOA	∞	∞	∞	∞	∞	∞ *1e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8801*	0/15
lmmIPOP400D	∞	∞	∞	∞	∞	∞ *8804*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	4722	7628	7666	7700	7758	1.4e5	9/15
IPOPsaACM	**1.9e4**(2e4)	∞	∞	∞	∞	∞ *2e7*	0/5
NEWUOA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *2e5*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8801*	0/15
lmmIPOP400D	∞	∞	∞	∞	∞	∞ *8813*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	41	41	41	41	41	41	15/15
IPOPsaACM	4.5(0.7)	5.1(0.7)	5.2(0.8)	5.2(0.8)	5.2(0.8)	5.2(0.8)	15/15
NEWUOA	**1.2**(0.1)$^{\star 4}$	**1.5**(0.4)$^{\star 4}$	**1.6**(0.5)$^{\star 3}$	**1.6**(0.5)$^{\star 3}$	**1.6**(0.5)$^{\star 3}$	**1.6**(0.5)$^{\star 3}$	15/15
BFGS	2.4(0.3)	2.7(0.5)	2.8(0.3)	2.8(0.3)	2.8(0.3)	2.8(0.3)	15/15
IPOP400D	5.4(1)	6.4(1)	6.5(1)	6.5(1)	6.5(1)	6.5(1)	15/15
lmmIPOP400D	5.0(1)	6.0(1)	6.1(1)	6.1(1)	6.1(1)	6.1(1)	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	1296	2343	3413	5220	6728	8409	15/15
IPOPsaACM	1.5(0.4)	1.2(0.3)	1.1(0.2)	1.1(0.2)	**1.2**(0.2)	1.3(0.3)	15/15
NEWUOA	**1**(0.3)*	**1**(0.4)	**1**(0.5)	**1.1**(0.5)	1.3(0.8)	**1.3**(0.7)	15/15
BFGS	3.6(2)	3.5(1)	3.4(1)	3.5(0.9)	3.6(0.7)	45(21)	0/15
IPOP400D	1.5(0.4)	1.2(0.2)	1.1(0.1)	1.1(0.1)	1.2(0.1)	16(17)	0/15
lmmIPOP400D	18(18)	56(56)	∞	∞	∞	∞ *8804*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	1351	4274	9503	16524	16583	16969	15/15
IPOPsaACM	1.0(1)	**1.6**(0.6)	**0.92**(0.6)	**0.66**(0.3)	**0.66**(0.3)	**0.65**(0.3)	15/15
NEWUOA	∞	∞	∞	∞	∞	∞ *5e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *2100*	0/15
IPOP400D	1.0(1)	29(35)	∞	∞	∞	∞ *8827*	0/15
lmmIPOP400D	**0.48**(0.1)	∞	∞	∞	∞	∞ *8819*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	2039	3871	4040	4219	4371	4484	15/15
IPOPsaACM	1.4(0.2)	1.3(0.1)	1.4(0.1)	1.4(0.1)	1.4(0.1)	1.4(0.1)	15/15
NEWUOA	**1**(0.3)$^{\star 2}$	**1**(0.6)	**1**(0.6)	**1**(0.5)	**1**(0.5)	**1**(0.5)	15/15
BFGS	1.8(0.3)	1.2(0.1)	1.2(0.1)	1.2(0.1)	1.2(0.1)	1.2(0.1)	15/15
IPOP400D	6.5(5)	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	1.7(0.5)	2.1(1)	2.2(1)	2.2(1)	2.2(1)	2.2(1)	12/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f9	1716	3102	3277	3455	3594	3727	15/15
IPOPsaACM	1.6(0.4)	1.8(1)	1.9(1)	1.9(1)	1.9(1.0)	1.9(1.0)	15/15
NEWUOA	**1.0**(0.2)$^{\star 3}$	**1**(0.6)$^{\star 2}$	**1**(0.6)$^{\star 2}$	**1**(0.5)$^{\star 2}$	**1**(0.5)$^{\star 2}$	**1**(0.5)$^{\star 2}$	15/15
BFGS	2.2(0.4)	2.2(1.0)	2.1(0.9)	2.0(0.9)	2.0(0.8)	1.9(0.8)	15/15
IPOP400D	4.5(0.8)	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	2.1(0.5)	2.1(0.2)	2.2(0.2)	2.4(0.2)	2.4(0.2)	2.7(1)	12/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f10	7413	8661	10735	14920	17073	17476	15/15
IPOPsaACM	**0.35**(0.1)$^{\star}_{\downarrow 4}$	**0.36**(0.1)$^{\star}_{\downarrow 4}$	**0.31**(0.0)$^{\star}_{\downarrow 4}$	**0.24**(0.0)$^{\star}_{\downarrow 4}$	**0.22**(0.0)$^{\star}_{\downarrow 4}$	**0.23**(0.0)$^{\star}_{\downarrow 4}$	15/15
NEWUOA	1.7(0.5)	2.6(0.8)	3.3(1)	4.0(0.8)	4.7(0.8)	5.8(1)	15/15
BFGS	1.0(0.2)	1(0.1)	1(0.5)	1.1(0.4)	3.1(4)	∞ *1e6*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	0.46(0.1)$_{\downarrow 4}$	0.47(0.1)$_{\downarrow 4}$	0.41(0.1)$_{\downarrow 4}$	0.33(0.0)$_{\downarrow 4}$	0.31(0.0)$_{\downarrow 4}$	0.32(0.0)$_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f11	1002	2228	6278	9762	12285	14831	15/15
IPOPsaACM	2.5(0.5)	1.2(0.2)	0.45(0.1)	**0.31**(0.1)	**0.26**(0.1)	**0.23**(0.0)	15/15
NEWUOA	15(2)	13(2)	5.8(0.6)	6.1(0.5)	6.6(0.3)	6.5(0.3)	15/15
BFGS	**1**(0.5)$^{\star 2}$	**1**(0.8)	1.3(0.7)	147(150)	∞	∞ *2e5*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	2.1(0.2)	1.1(0.1)	**0.43**(0.0)	0.31(0.0)$_{\downarrow 4}$	0.28(0.0)$_{\downarrow 4}$	0.25(0.0)$_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f12	1042	1938	2740	4140	12407	13827	15/15
IPOPsaACM	**0.67**(0.1)	**0.88**(0.7)	**1.1**(0.7)	**1.2**(0.5)	**0.56**(0.2)$_{\downarrow 2}$	**0.60**(0.2)$_{\downarrow 2}$	15/15
NEWUOA	3.0(3)	3.0(2)	3.0(2)	2.5(1)	1(0.4)	1(0.4)	15/15
BFGS	1.6(0.9)	1.6(2)	1.6(1)	1.6(1)	1.8(2)	45(56)	1/15
IPOP400D	2.3(1)	3.1(2)	8.0(9)	32(36)	∞	∞ *8811*	0/15
lmmIPOP400D	1.1(0.1)	1.1(0.7)	1.3(0.9)	1.7(1)	0.92(0.7)	2.2(2)	4/15

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f13	652	2021	2751	18749	24455	30201	15/15
IPOPsaACM	1.7(2)	1.7(0.8)	1.5(0.7)	0.34(0.2)$_{\downarrow 4}$	0.37(0.1)$_{\downarrow 4}$	0.41(0.2)$_{\downarrow 4}$	15/15
NEWUOA	**1**(1)*	3.0(5)	9.3(12)	19(24)	∞	∞ *2e5*	0/15
BFGS	1.7(0.2)	**1**(0.0)	1(0.0)	23(25)	∞	∞ *5e5*	0/15
IPOP400D	4.2(5)	3.5(4)	6.6(7)	∞	∞	∞ *8813*	0/15
lmmIPOP400D	2.0(0.2)	1.0(0.2)	**0.99**(0.2)	**0.22**(0.0)$^{\star}_{\downarrow 4}$	**0.21**(0.0)$^{\star}_{\downarrow 4}$	**0.21**(0.0)$^{\star}_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f14	75	239	304	932	1648	15661	15/15
IPOPsaACM	3.0(0.6)	1.8(0.3)	1.9(0.4)	1.4(0.2)	1.4(0.1)	**0.23**(0.0)$^{\star}_{\downarrow 4}$	15/15
NEWUOA	**1.5**(0.7)*	**1**(0.3)$^{\star 2}$	**1**(0.3)$^{\star 3}$	**1**(0.2)	9.1(0.9)	43(32)	0/15
BFGS	2.7(1.0)	1.8(0.7)	2.0(0.7)	1.2(0.2)	**1.1**(0.2)$^{\star 2}$	∞ *2e5*	0/15
IPOP400D	2.5(1)	2.2(0.4)	3.1(0.3)	3.9(0.3)	∞	∞ *8811*	0/15
lmmIPOP400D	3.0(1)	2.2(0.4)	2.8(0.4)	2.1(0.1)	2.2(0.1)	0.35(0.0)$_{\downarrow 4}$	15/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f15	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
IPOPsaACM	**0.60**(0.5)	**0.92**(0.7)	**0.53**(0.4)	**0.52**(0.4)	**0.37**(0.3)$_{\downarrow 4}$	**0.37**(0.3)$_{\downarrow 4}$	15/15
NEWUOA	∞	∞	∞	∞	∞	∞ *1e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	4.3(4)	∞	∞	∞	∞	∞ *8807*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f16	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
IPOPsaACM	1.8(0.8)	**0.55**(0.4)	**0.77**(0.8)	**0.52**(0.4)	**0.55**(0.4)	**0.50**(0.3)$_{\downarrow}$	15/15
NEWUOA	16(17)	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *3e5*	0/15
IPOP400D	1.6(0.5)	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	**1.2**(0.4)	0.75(0.8)	∞	∞	∞	∞ *8809*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f17	63	1030	4005	30677	56288	80472	15/15
IPOPsaACM	2.5(2)	0.91(0.3)	0.98(1)	**1.2**(0.5)	**1.2**(0.4)	**1.1**(0.5)	15/15
NEWUOA	16(4)	∞	∞	∞	∞	∞ *2e6*	0/15
BFGS	359(613)	∞	∞	∞	∞	∞ *4e5*	0/15
IPOP400D	**0.79**(0.7)	**0.80**(0.2)	**0.80**(1)	∞	∞	∞ *8811*	0/15
lmmIPOP400D	1(1)	1.4(0.6)	0.80(1)	4.1(5)	∞	∞ *8805*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f18	621	3972	19561	67569	1.3e5	1.5e5	15/15
IPOPsaACM	0.96(0.5)	1.4(2)	**0.91**(0.6)	**0.78**(0.5)	**0.88**(0.4)	**1.3**(0.8)	15/15
NEWUOA	1.2e4(1e4)	∞	∞	∞	∞	∞ *2e6*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *4e5*	0/15
IPOP400D	**0.74**(0.3)$_{\downarrow}$	**0.67**(0.3)	3.1(3)	∞	∞	∞ *8811*	0/15
lmmIPOP400D	0.76(0.3)	1.5(2)	6.4(8)	∞	∞	∞ *8804*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f19	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
IPOPsaACM	154(50)	3.0e4(2e4)	**0.61**(0.5)	**0.33**(0.1)	**0.32**(0.2)	**0.32**(0.2)	14/15
NEWUOA	76(50)	4.3e6(5e6)	∞	∞	∞	∞ *2e6*	0/15
BFGS	1.2e6(1e6)	∞	∞	∞	∞	∞ *2e5*	0/15
IPOP400D	**1**(0)	**1**(0)	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	**1**(0)	**1**(0)	∞	∞	∞	∞ *8805*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f20	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
IPOPsaACM	2.8(0.5)	**1.7**(0.8)	**0.49**(0.2)$_{\downarrow 4}$	**0.45**(0.2)$_{\downarrow 4}$	**0.45**(0.2)$_{\downarrow 4}$	**0.45**(0.2)$_{\downarrow 4}$	15/15
NEWUOA	**1**(0.5)$^{\star 4}$	15(19)	∞	∞	∞	∞ *4e5*	0/15
BFGS	2.1(0.4)	5.8(5)	∞	∞	∞	∞ *4e5*	0/15
IPOP400D	3.8(1)	∞	∞	∞	∞	∞ *8801*	0/15
lmmIPOP400D	3.6(0.6)	∞	∞	∞	∞	∞ *8820*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f21	561	6541	14103	14643	15567	17589	15/15
IPOPsaACM	2.6(4)	53(94)	157(308)	151(297)	142(279)	126(247)	15/15
NEWUOA	1.7(3)	2.2(2)	**1.2**(2)	**1.2**(2)	**1.1**(2)	**1**(1)	15/15
BFGS	1.9(4)	5.5(6)	4.6(6)	4.5(5)	4.3(5)	7.3(9)	2/15
IPOP400D	5.8(8)	5.7(6)	4.3(5)	4.2(5)	4.0(4)	3.5(4)	2/15
lmmIPOP400D	**1.7**(2)	**1.8**(2)	1.4(2)	1.4(1)	1.3(2)	1.2(2)	5/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f22	467	5580	23491	24948	26847	1.3e5	12/15
IPOPsaACM	175(98)	978(1807)	∞	∞	∞	∞ *2e7*	0/15
NEWUOA	**1**(1)*	4.9(6)	**6.8**(8)	**6.4**(7)	**6.0**(7)	**1.2**(1)	7/15
BFGS	2.5(2)	**1.8**(2)	8.1(8)	7.7(9)	10(10)	14(16)	0/15
IPOP400D	11(15)	10(12)	∞	∞	∞	∞ *8801*	0/15
lmmIPOP400D	6.7(10)	4.7(5)	∞	∞	∞	∞ *8811*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f23	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
IPOPsaACM	4.3(6)	2.9e4(3e4)	906(1102)	∞	∞	∞ *2e7*	0/15
NEWUOA	12(8)	**3.5**(3)$^{\star 3}$	**32**(40)	∞	∞	∞ *2e5*	0/15
BFGS	47(26)	304(326)	∞	∞	∞	∞ *1e5*	0/15
IPOP400D	2.5(3)	81(90)	∞	∞	∞	∞ *8801*	0/15
lmmIPOP400D	**1.6**(2)	∞	∞	∞	∞	∞ *8823*	0/15
Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f24	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
IPOPsaACM	**28**(30)	**38**(44)	∞	∞	∞	∞ *2e7*	0/15
NEWUOA	∞	∞	∞	∞	∞	∞ *2e5*	0/15
BFGS	∞	∞	∞	∞	∞	∞ *1e5*	0/15
IPOP400D	∞	∞	∞	∞	∞	∞ *8811*	0/15
lmmIPOP400D	∞	∞	∞	∞	∞	∞ *8817*	0/15

Table 2: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values in dimension 20. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in *italics*, if $\text{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. Best results are printed in bold.