S. Krajči (ed.): ITAT 2018 Proceedings, pp. 72–79
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Zbyněk Pitra, Jakub Repický, and Martin Holeňa

Boosted Regression Forest for the Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy

Zbyněk Pitra[1,2], Jakub Repický[1,3], and Martin Holeňa[1]

[1] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
{pitra, repicky, holena}@cs.cas.cz
[2] Faculty of Nuclear Sciences and Physical Engineering, CTU in Prague
Břehová 7, 115 19 Prague 1, Czech Republic
[3] Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Prague 1, Czech Republic

Abstract: Many real-world problems belong to the area of continuous black-box optimization, where evolutionary optimizers have become very popular inspite of the fact that such optimizers require a great amount of real-world fitness function evaluations, which can be very expensive or time-consuming. Hence, regression surrogate models are often utilized to evaluate some points instead of the fitness function. The Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy (DTS-CMA-ES) is a surrogate-assisted version of the state-of-the-art continuous black-box optimizer CMA-ES using Gausssian processes as a surrogate model to predict the whole distribution of the fitness function. In this paper, the DTS-CMA-ES is studied in connection with the boosted regression forest, another regression model capable to estimate the distribution. Results of testing regression forest and Gaussian processes, the former in 20 different settings, as a surrogate models in the DTS-CMA-ES on the set of noiseless benchmarks are reported.

1 Introduction

Real-world problems can be very costly in terms of various resources, most often money and time. An important kind of such problems are optimization tasks in which the objective function cannot be expressed mathematically, but has to be evaluated empirically, through measurements, experiments, or simulations. Such optimization tasks are called *black-box*. *Evolutionary algorithms* have become very successful in this optimization field. In case of limited resources, the number of empirical evaluations necessary to achieve the target distance to the optimal value by the optimization algorithm should be as small as possible.

Nowadays, the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [12] is considered to be the state-of-the-art algorithm for continuous black-box optimization. On the other hand, the CMA-ES can require many function evaluations to achieve the target distance from the optimum in problems where the fitness function is expensive. Therefore, several surrogate-assisted versions of the CMA-ES have been presented during the last decade (an overview can be found in [2, 20]). Such CMA-ES based algorithms save expensive evaluations through utilization of a regression surrogate model instead of the original function on a selected part of the current population.

The local meta-model CMA-ES (lmm-CMA-ES) was proposed in [16] and later improved in [1]. The algorithm constructs local quadratic surrogate models and controls changes in population ranking after each evaluation of the original fitness.

In [18], the *Doubly Trained Surrogate CMA-ES* (DTS-CMA-ES) was presented. It utilizes the ability of *Gaussian Processes* (GPs) [21] to estimate the whole distribution of fitness function values to select most promising points to be evaluated by the original fitness. The DTS-CMA-ES was also tested [19] in a version where metric GPs were replaced by ordinal GPs inspired by the fact that the CMA-ES is invariant with respect to order preserving transformations. However, up to our knowledge, there has been no research into combining the DTS-CMA-ES with the surrogate model capable to predict the whole probability distribution of fitness values, where the model was not based on GPs. The ensembles of regression trees [4] are also able to estimate the whole distribution of values. They have been already utilized as surrogate models for the CMA-ES in [3] using different evolution control strategy than the DTS-CMA-ES.

In this paper, we use ensembles of regression trees as surrogate models in the DTS-CMA-ES algorithm. Due to an increasing popularity of gradient boosting [10], we train the ensembles of regression trees, i. e., *regression forests* (RFs), using such strategy. Up to our knowledge, this is the first time the boosted RF regression is utilized for surrogate modeling in the DTS-CMA-ES context. Therefore, we investigate also the suitability of several different settings of the employed regression method to this end. We experimentaly compare the original DTS-CMA-ES with a new version using RFs together with the original CMA-ES on the noiseless part of the *Comparing-Continuous-Optimizers* (COCO) platform [13, 14] in the expensive settings with the limited budget of fitness evaluations.

The rest of the paper is organized as follows. Section 2 describes the DTS-CMA-ES algorithm. Section 3 gives a short introduction into gradient boosting and regression

Algorithm 1 DTS-CMA-ES [18]

Input: λ (population-size), y_{target} (target value), f (original fitness function), n_{orig} (number of original-evaluated points), $\mathcal{C}$ (uncertainty criterion)

$\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES initialize
$\mathcal{A} \leftarrow \varnothing$ {*archive initialization*}
while $\min\{y|(\mathbf{x},y) \in \mathcal{A}\} > y_{\text{target}}$ **do**
  $\{\mathbf{x}_k\}_{k=1}^{\lambda} \sim \mathcal{N}\left(\mathbf{m}, \sigma^2\mathbf{C}\right)$ {*CMA-ES sampling*}
  $f_{\mathcal{M}1} \leftarrow \text{trainModel}(\mathcal{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model training*}
  $(\hat{y}, \mathbf{s}^2) \leftarrow f_{\mathcal{M}1}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {*model evaluation*}
  $\mathbf{X}_{\text{orig}} \leftarrow$ select n_{orig} best points according to $\mathcal{C}(\hat{y}, \mathbf{s}^2)$
  $\mathbf{y}_{\text{orig}} \leftarrow f(\mathbf{X}_{\text{orig}})$ {*original fitness evaluation*}
  $\mathcal{A} = \mathcal{A} \cup \{(\mathbf{X}_{\text{orig}}, \mathbf{y}_{\text{orig}})\}$ {*archive update*}
  $f_{\mathcal{M}2} \leftarrow \text{trainModel}(\mathcal{A}, \sigma, \mathbf{m}, \mathbf{C})$ {*model retrain*}
  $\mathbf{y} \leftarrow f_{\mathcal{M}2}([\mathbf{x}_1, \ldots, \mathbf{x}_\lambda])$ {2^{nd} *model prediction*}
  $(\mathbf{y})_k \leftarrow y_{\text{orig},i}$ for all original-evaluated $y_{\text{orig},i} \in \mathbf{y}_{\text{orig}}$
  $\sigma, \mathbf{m}, \mathbf{C} \leftarrow$ CMA-ES update $(\mathbf{y}, \sigma, \mathbf{m}, \mathbf{C})$
end while
$\mathbf{x}_{\text{res}} \leftarrow \mathbf{x}_k$ from $\mathcal{A}$ where y_k is minimal

Output: $\mathbf{x}_{\text{res}}$ (point with minimal y)

tree algorithms. Section 4 contains experimental setup and results. Section 5 concludes the paper and discusses future research.

2 Doubly Trained Surrogate CMA-ES

The DTS-CMA-ES, introduced in [18], is a modification of the CMA-ES, where the ability of GPs to estimate the whole distribution of fitness function is utilized to select the most promising points out of the sampled population. The points selected using some uncertainty criterion $\mathcal{C}$ are evaluated with the original fitness function f and included into the set of points employed for the GP model retraining. The remaining points from the population are reevaluated using the retrained GP model. The core CMA-ES parameters (σ, $\mathbf{m}$, $\mathbf{C}$, etc.) are computed according to the original CMA-ES algorithm. The DTS-CMA-ES pseudocode is shown in Algorithm 1.

The only models capable to predict the whole fitness distribution used so far in connection with the DTS-CMA-ES were based on GPs.

3 Boosted regression forest

Regression forest [5] is an ensemble of regression decision trees [4]. In the last decade, the *gradient tree boosting* [10] has become very popular and successful technique for forest learning. Therefore, we will focus only on this method.

3.1 Gradient boosted regression trees

Let's consider binary regression trees, where each observation $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$ passes through a series of binary split functions s associated with internal nodes and arrives in the leaf node containing a real-valued constant trained to be the prediction of an associated function value y. A binary split function determines whether $\mathbf{x}$ proceeds to its left or right child of the respective node.

The gradient boosted forest has to be trained in an additive manner. Let $\hat{y}_i^{(t)}$ be the prediction of the i-th point of the t-th tree. The t-th tree f_t is obtained in the t-th iteration of the boosting algorithm through optimization of the following regularized objective function:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t), \qquad (1)$$

$$\text{where} \quad \Omega(f) = \gamma T_f + \frac{1}{2}\lambda \left\|w_f\right\|^2,$$

l is a differentiable convex loss function $l : \mathbb{R}^2 \to \mathbb{R}$, T_f is the number of leaves in a tree f, and w_f are weights of its individual leaves. The regularization term Ω is used to control model complexity through penalization constants $\gamma \geq 0$ and $\lambda \geq 0$.

The tree growing starts with one node (root) and a set of all input data. Individual branches are then recursively added according to the gain of split considering splitting data $\mathcal{S}_{L+R}$ into sets $\mathcal{S}_L$ (left branch set) and $\mathcal{S}_R$ (right branch set). The gain can be derived using (1) as follows (see [7] for details):

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[r(\mathcal{S}_L) + r(\mathcal{S}_R) - r(\mathcal{S}_{L+R})\right] - \gamma,$$
$$r(\mathcal{S}) = \frac{\left(\sum_{y\in\mathcal{S}} g(y)\right)^2}{\sum_{y\in\mathcal{S}} h(y) + \lambda}, \qquad (2)$$

where $g(y) = \partial_{\hat{y}^{(t-1)}} l(y, \hat{y}^{(t-1)})$ and $h(y) = \partial^2_{\hat{y}^{(t-1)}} l(y, \hat{y}^{(t-1)})$ are the first and second order derivatives of the loss function.

The tree growing is stopped when one of the user-defined conditions is satisfied, e. g., the tree reaches the maximum number of levels, or no node can be split without dropping the number of points in any node under the allowed minimum.

The overall boosted forest prediction is obtained through averaging individual tree predictions, where each leaf j in a t-th tree has weight

$$w_j^{(t)} = -\frac{\sum_{y\in\mathcal{S}_j} g(y)}{\sum_{y\in\mathcal{S}_j} h(y) + \lambda}, \qquad (3)$$

where S_j is the set of all training inputs that end in the leaf j. As a prevention of overfitting, the random subsampling of input features or input points can be employed.

3.2 Split algorithms

The decision split function s can be found through numerous algorithms developed since the original CART [4]. In the following paragraphs, we survey some of them.

Traditional CART [4] are based on searching *axis-parallel* hyperplanes. To find the splitting hyperplane, the value of each training point $\mathbf{x} = (x^{(1)}, \ldots, x^{(D)})$ in dimension $\mathbf{x}^{(d)}$, $d \in \{1, \ldots, D\}$ is considered as a threshold for the dimension d defining the candidate hyperplane. The full-search through all dimensions is done and the splitting hyperplane with the highest gain is selected.

In SECRET [9], the expectation-maximization algorithm for *Gaussian mixtures* is utilized to find two clusters and the regression task is transformed into classification based on assignments of points to these clusters. Splitting oblique hyperplane is provided through linear or quadratic discriminant analysis.

Deterministic *hill-climbing* with effective randomization is employed to find a most suitable linear hyperplane in the algorithm OC1 [17]. Split-finding starts with a random hyperplane or with a good axis-parallel hyperplane found similarly to CART. Then the hyperplane's direction is deterministically perturbated in each axis to maximize the split gain. Once no improvement is possible, a number of random jumps is performed as an attempt to escape from local optima. In case of successful random jump, deterministic perturbation is performed again.

In [15] (PAIR), the *pairs of points* are used to define a projection for splitting the input space. For each pair of points, a normal vector defining a direction is constructed. The rest of training points is projected onto this vector and the projected values are utilized as thresholds defining splitting hyperplanes orthogonal to constructed normal vector. To reduce complexity, only the threshold halfway between the defining pair can be considered.

A nonparametric function estimation method called SUPPORT [6] is based on the analysis of *residuals* after regression to find a split. At the beginning, polynomial regression is performed on the training data. The points under the curve (negative residuals) present the first class, and the rest of points (positive or zero residuals) presents the second class. Afterwards, distribution analysis is applied to find a split.

4 Experimental evaluation

In this section, we compare the performances of the DTS-CMA-ES using the RFs as a surrogate model in several different settings to the original DTS-CMA-ES version, the original CMA-ES, and the lmm-CMA-ES on the noiseless part of the COCO platform [13, 14].

4.1 Experimental setup

The considered algorithms were compared on 24 noiseless single-objective continuous benchmark functions from the COCO testbed [13, 14] in dimensions $D = 2, 3, 5$, and 10 on 15 different instances per function. Each algorithm had a budget of $250D$ function evaluations to reach the target distance $\Delta f_T = 10^{-8}$ from the function optimum. The parameter settings of the tested algorithms are summarized in the following paragraphs.

The original CMA-ES was employed in its IPOP-CMA-ES version (Matlab code v. 3.61) with the following settings: the number of restarts = 4, IncPopSize = 2, $\sigma_{\text{start}} = \frac{8}{3}$, $\lambda = 4 + \lfloor 3 \log D \rfloor$. The remaining settings were left default.

The lmm-CMA-ES was utilized in its improved version published in [1]. The results have been downloaded from the COCO results data archive[1] in its GECCO 2013 settings.

The original DTS-CMA-ES was tested using the overall best settings from [2]: the probability of improvement as the uncertainty criterion, the population size $\lambda = 8 + \lfloor 6 \log D \rfloor$, and the number of originally-evaluated points $n_{\text{orig}} = \lceil 0.05\lambda \rceil$.

Considering decision tree settings, the five splitting methods from the following algoritms were employed: CART [4], SECRET [9], OC1 [17], SUPPORT [6], and PAIR [15]. Due to the different properties of individual splitting methods, the number of $\mathcal{L}_{\text{split}}$ evaluations was limited to $10D$ per node to restrict the algorithms which test a great number of hyperplanes. For the same reason, the number of tresholds generated by a projection of points to a hyperplane was set to 10 quantile-based values in CART, OC1, and to a median value in PAIR, and the searching an initial axis-aligned hyperplane in OC1 was limited to $\lceil \frac{10D}{3} \rceil$ $\mathcal{L}_{\text{split}}$ evaluations.

The RFs as a surrogate model were tested using the gradient boosting ensemble method. The maximum tree depth was set to 8, in accordance with [7]. In addition, the number of trees n_{tree}, the number of points N_t bootstrapped out of N archive points, and the number of randomly subsampled dimensions used for training the individual tree n_D were sampled from the values in Table 1.

The DTS-CMA-ES in combination with RFs was tested with the following settings: the probability of improvement as the uncertainty criterion, the population size $\lambda = 8 + \lfloor 6 \log D \rfloor$, and the number of originally-evaluated points n_{orig} with 4 different values $\lceil 0.05\lambda \rceil$, $\lceil 0.1\lambda \rceil$, $\lceil 0.2\lambda \rceil$, and $\lceil 0.4\lambda \rceil$. The rest of DTS-CMA-ES parameters have been taken identical to the overall best settings from [2].

4.2 Results

Result from experiments are presented in Figures 1–4 and also in Table 2. The graphs in Figures 1–4 depict the scaled best-achieved logarithms $\Delta_f^{\log}$ of median distances Δ_f^{med} to the functions optimum for the respective number of function evaluations per dimension (FE/D). Medians Δ_f^{med} (and in Figure 1 also 1st and 3rd quartiles) are calculated from 15 independent instances for each respective algorithm, function, and dimension. The scaled logarithms of Δ_f^{med} are calculated as

[1] `http://coco.gforge.inria.fr/data-archive/2013/lmm-CMA-ES_auger_noiseless.tgz`

Table 1: Experimental settings of RF: n_{orig} – number of originally-evaluated points , n_{tree} – number of trees in RF, N_t, n_D – number of tree points and dimensions. Split methods and n_{orig} are selected using full-factorial design, n_{tree}, N_t, and n_D are sampled.

parameter	values
n_{orig}	$\{\lceil 0.05\lambda \rceil, \lceil 0.1\lambda \rceil, \lceil 0.2\lambda \rceil, \lceil 0.4\lambda \rceil\}$
split	{CART, SECRET, OC1, SUPPORT, PAIR}
n_{tree}	$\{64, 128, 256, 512, 1024\}$
N_t	$\lceil \{0.25, 0.5, 0.75, 1\} \cdot N \rceil$
n_D	$\lceil \{0.25, 0.5, 0.75, 1\} \cdot D \rceil$

$$\Delta_f^{\log} = \frac{\log \Delta_f^{\text{med}} - \Delta_f^{\text{MIN}}}{\Delta_f^{\text{MAX}} - \Delta_f^{\text{MIN}}} \log_{10}\left(1/10^{-8}\right) + \log_{10} 10^{-8}$$

where Δ_f^{MIN} (Δ_f^{MAX}) is the minimum (maximum) $\log \Delta_f^{\text{med}}$ found among all the compared algorithms for the particular function f and dimension D between 0 and 250 FE/D. Such scaling enables the aggregation of $\Delta_f^{\log}$ graphs across arbitrary number of functions and dimensions (see Figures 2, 3, and 4). The values are scaled to the $[-8, 0]$ interval, where -8 corresponds to the minimal and 0 to the maximal distance. This visualization was chosen due to better ability to distinguish the differences in the convergence of tested algorithms in comparison with the default visualization used by the COCO platform.

We compare the statistical significance of differences in algorithms' performance on 24 COCO functions in $5D$ for separately two evaluation budgets utilizing the Iman and Davenport's improvement of the Friedman test [8]. Let $\#\text{FE}_\text{T}$ be the smallest number of FE on which at least one algorithm reached the target distance, i. e., satisfied $\Delta_f^{\text{med}} \leq \Delta f_T$, or $\#\text{FE}_\text{T} = 250D$ if no algorithm reached the target within $250D$ evaluations. The algorithms are ranked on each function with respect to Δ_f^{med} at a given budget of FE. The null hypothesis of equal performance of all algorithms is rejected for the higher function evaluation budget $\#\text{FEs} = \#\text{FE}_\text{T}$ ($p < 10^{-3}$), as well as for the lower budget $\#\text{FEs} = \frac{\#\text{FE}_\text{T}}{3}$ ($p < 10^{-3}$).

We test pairwise differences in performance utilizing the post-hoc test to the Friedman test [11] with the Bergmann-Hommel correction controlling the family-wise error. The numbers of functions at which one algorithm achieved a higher rank than the other are enlisted in Table 2. The table also contains the pairwise statistical significances.

The graphs in Figures 2 and 3 summarize the performance of five different split algorithms and four n_{orig} values from twenty different settings respectively. We found that the convergence of DTS-CMA-ES is quite similar regardless the split algorithm with slightly better results of SECRET and SUPPORT – the algorithms utilizing classification methods to find the splitting hyperplane between previously created clusters of training points. The results also show that lower n_{orig} values provide better performance in the initial phase of the optimization run and higher values are more successful starting from the 100-150 FE/D. Due to the presented results, the following comparisons contain the performances of the DTS-CMA-ES with $n_{\text{orig}} = \lceil 0.4\lambda \rceil$ in combination with RF using SECRET and SUPPORT as split algorithms.

As can be seen in Figures 1 and 4, the performance of RFs is considerably worse than the performance of GPs in combination with the DTS-CMA-ES and better than the performance of the original CMA-ES. RF model provides faster convergence from approximitely 100 FE/D on the regularly multimodal Rastrigin functions (f_3, f_4, and f_{15}) where the RF apparently does not prevent the original CMA-ES from exploiting the global structure of a function. The performance of RF-DTS-CMA-ES is noticably lower especially on the elipsoid (f_1, f_2, f_7, and f_{10}), Rastrigin (f_8, f_9), and ill-condition functions (f_{11-14}), where smooth models are much more convenient for regression. On the other hand, RFs help the CMA-ES to convergence especially on the multimodal functions f_{16-19}, where the performance of RF-DTS-CMA-ES is the best of all compared algorithms.

5 Conclusions & Future work

In this paper, we have compared the RF model using gradient boosting as the ensemble method with the GP regression model, both used as surrogate models in the DTS-CMA-ES algorithm. Different methods of space splitting in regression trees were investigated.

The split algorithms SECRET and SUPPORT based on the classification of the input points provide slightly better performance as to the CMA-ES convergence than the other algorithms tested. Moreover, the performance of DTS-CMA-ES using RFs differs according to the number of originally-evaluated points: the lower their number, the sooner the algorithm converges, possibly to a local optimum, which makes convergence to the global one more difficult. We found that the RF model usually reduces the number of fitness evaluations required by the CMA-ES, especially on multi-modal functions, where the provided speed-up was the best among all compared algorithms for a number of evaluations higher than approximitely 110 FE/D.

A possible perspective for future research is to improve RF models by implementing non-constant (linear, quadratic) models to regression tree leaves, which could make the RFs prediction more convenient for smooth functions. Investigation of other split algorithms could also bring interesting results. Another perspective for future research is an automatical selection of the most convenient surrogate model for the CMA-ES inside the algorithm itself.

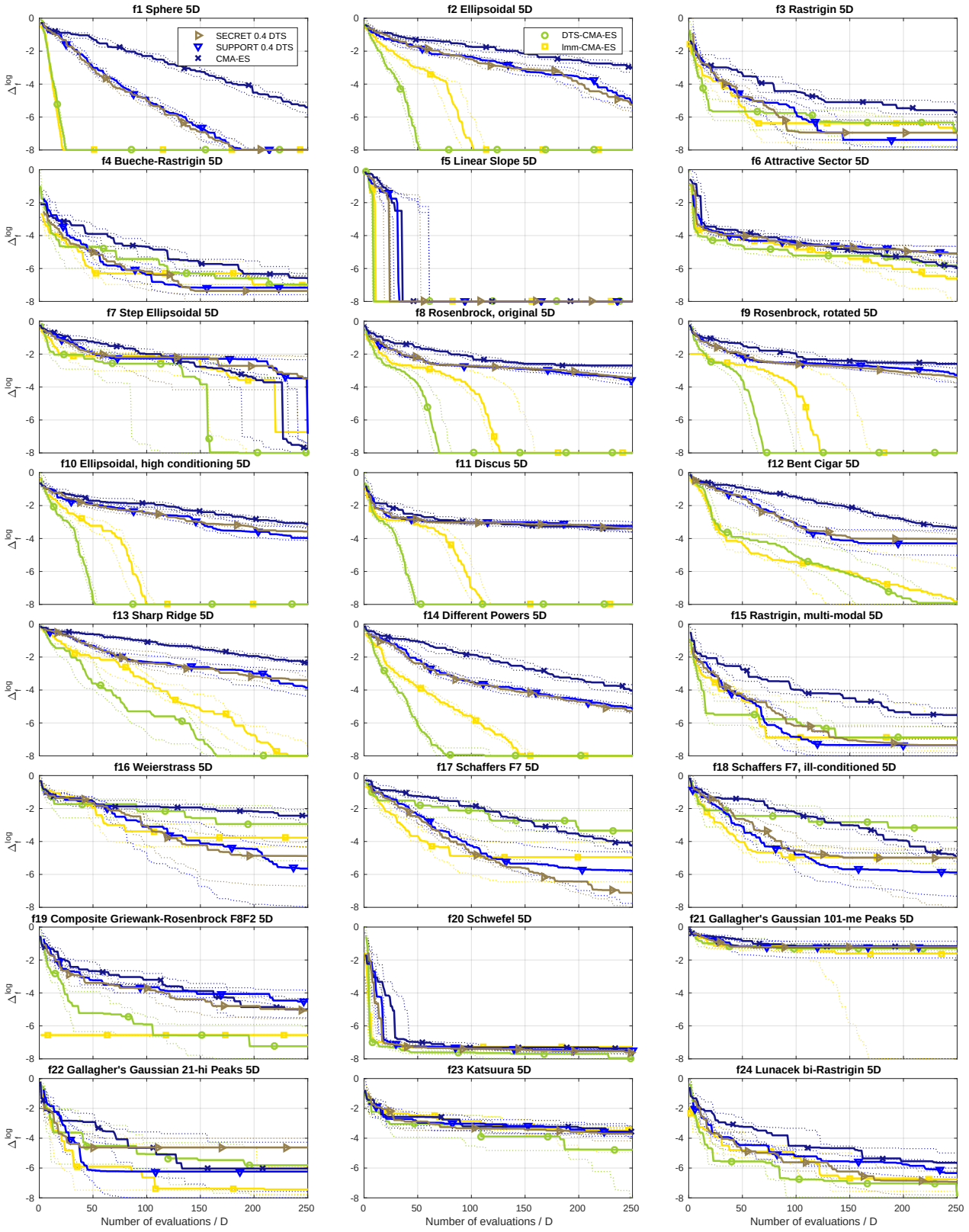

Figure 1: Medians (solid) and 1st/3rd quartiles (dotted) of the distances to the optima of 24 COCO benchmarks in 5D for algorithms CMA-ES, DTS-CMA-ES, lmm-CMA-ES, and 2 RF settings of DTS-CMA-ES. Medians/quartiles were calculated across 15 independent instances for each algorithm and are shown in the $\log_{10}$ scale.

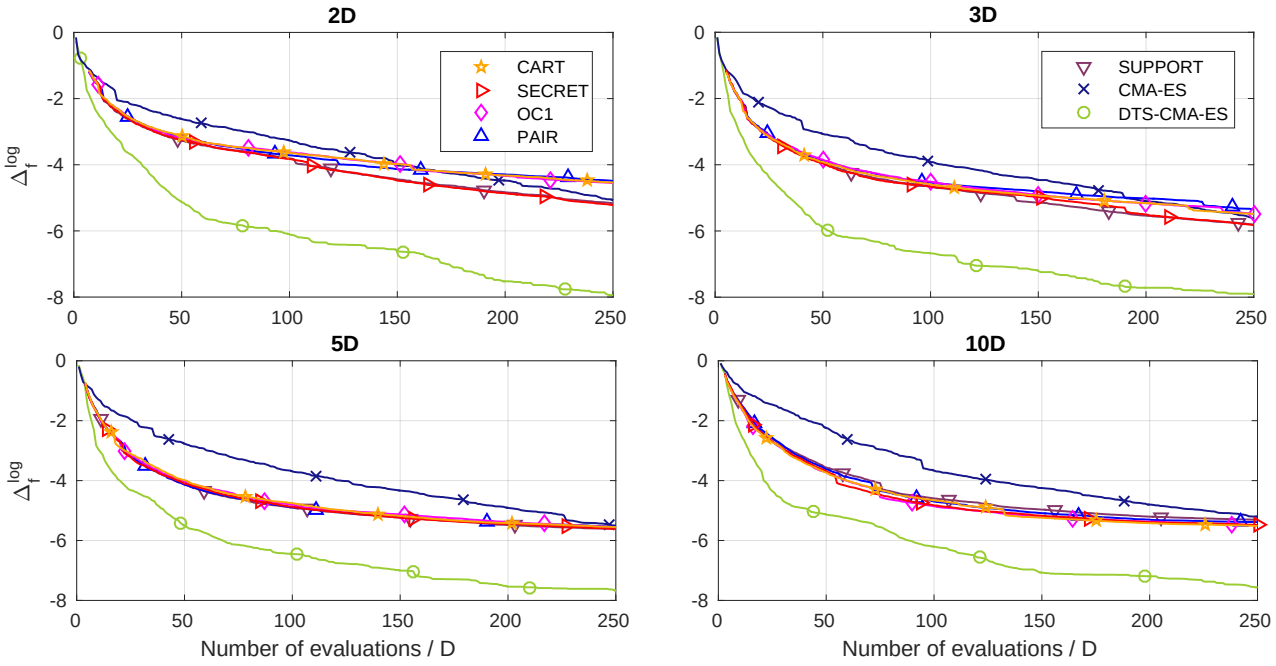

Figure 2: Scaled median distances $\Delta_f^{\log}$ of decision tree split algorithms averaged over all 24 COCO functions in 2D, 3D, 5D, and 10D for algorithms CART, SECRET, OC1, PAIR, and SUPPORT in combination with the DTS-CMA-ES and all tested numbers of originally-evaluated points.

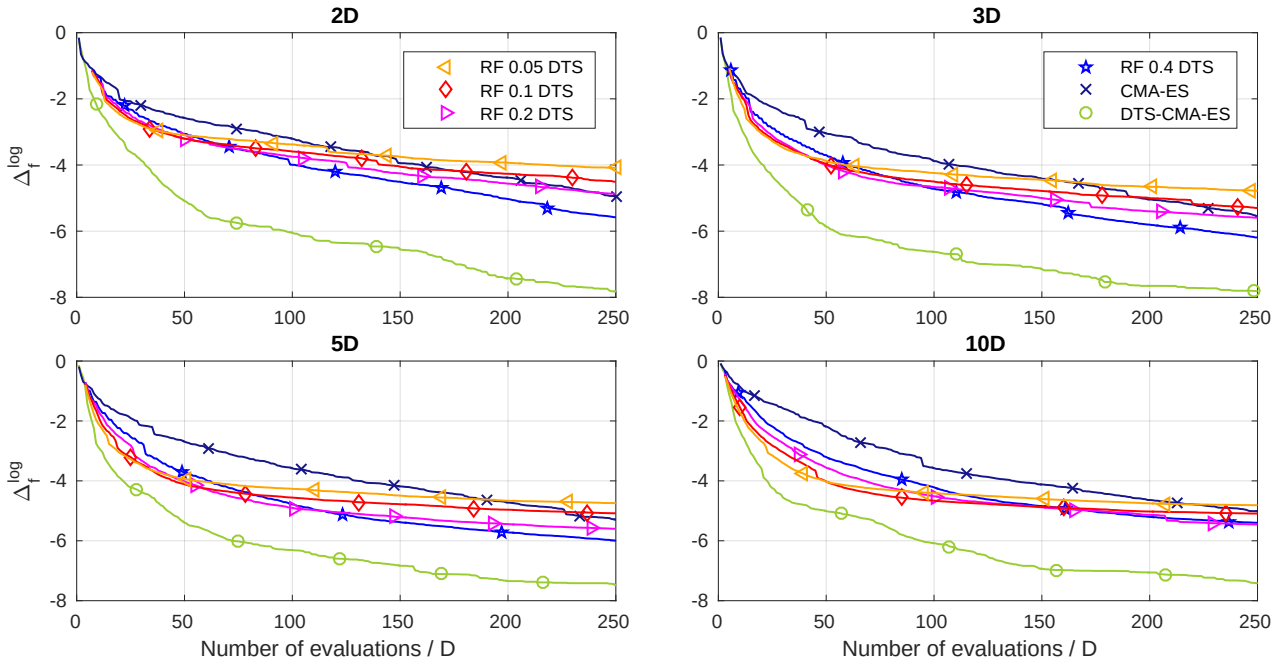

Figure 3: Scaled median distances $\Delta_f^{\log}$ of the DTS-CMA-ES with RFs comparing different numbers of originally-evaluated points averaged over all 24 COCO functions in 2D, 3D, 5D, and 10D for values $\lceil 0.05\lambda \rceil$, $\lceil 0.1\lambda \rceil$, $\lceil 0.2\lambda \rceil$, and $\lceil 0.4\lambda \rceil$ summarized accross all tested splitting algorithms.

Table 2: A pairwise comparison of the algorithms in 5*D* over the COCO for different evaluation budgets. The number of wins of *i*-th algorithm against *j*-th algorithm over all benchmark functions is given in *i*-th row and *j*-th column. The asterisk marks the row algorithm being significantly better than the column algorithm according to the Friedman post-hoc test with the Bergmann-Hommel correction at family-wise significance level $\alpha = 0.05$.

5D	SECRET 0.4 DTS		SUPPORT 0.4 DTS		CMA-ES		DTS-CMA-ES		lmm-CMA-ES	
$^{\#FEs}/_{\#FE_T}$	1/3	1	1/3	1	1/3	1	1/3	1	1/3	1
SECRET 0.4 DTS	—	—	11.5	11	23*	21*	8	6	5	8
SUPPORT 0.4 DTS	12.5	13	—	—	24*	21*	7	7	7	8
CMA-ES	1	3	0	3	—	—	3	4	1	3
DTS-CMA-ES	16	18	17	17	21*	20*	—	—	14	14
lmm-CMA-ES	19	16	17	16	23*	21*	10	10	—	—

Acknowledgements

The reported research was supported by the Czech Science Foundation grant No. 17-01251, by the Grant Agency of the Czech Technical University in Prague with its grant No. SGS17/193/OHK4/3T/14, and by Specific College Research project number 260 453. Further, access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] A. Auger, D. Brockhoff, and N. Hansen. Benchmarking the local metamodel CMA-ES on the noiseless BBOB'2013 test bed. In *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, Companion Material Proceedings*, pages 1225–1232, 2013.

[2] L. Bajer. *Model-based evolutionary optimization methods*. PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, 2018.

[3] L. Bajer, Z. Pitra, and M. Holeňa. Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1143–1150, New York, NY, USA, July 2015. ACM.

[4] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4(1):143–167, 1994.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. KDD '16, pages 785–794. ACM, 2016.

[8] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[9] A. Dobra and J. Gehrke. SECRET: A scalable linear regression tree algorithm. KDD '02, pages 481–487. ACM, 2002.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[11] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[12] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation*, number 192, pages 75–102. Springer, 2006.

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

[14] 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.

[15] G. E. Hinton and M. Revow. Using pairs of data-points to define splits for decision trees. In *Advances in Neural Information Processing Systems*, volume 8, pages 507–513. MIT Press, 1996.

[16] S. Kern, N. Hansen, and P. Koumoutsakos. Local Metamodels for Optimization Using Evolution Strategies. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 939–948. Springer Berlin Heidelberg, 2006.

[17] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *J. Artif. Int. Res.*, 2(1):1–32, 1994.

[18] Z. Pitra, L. Bajer, and M. Holeňa. Doubly trained evolution control for the Surrogate CMA-ES. In *PPSN XIV Proceedings*, pages 59–68. Springer, 2016.

[19] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Ordinal ver-

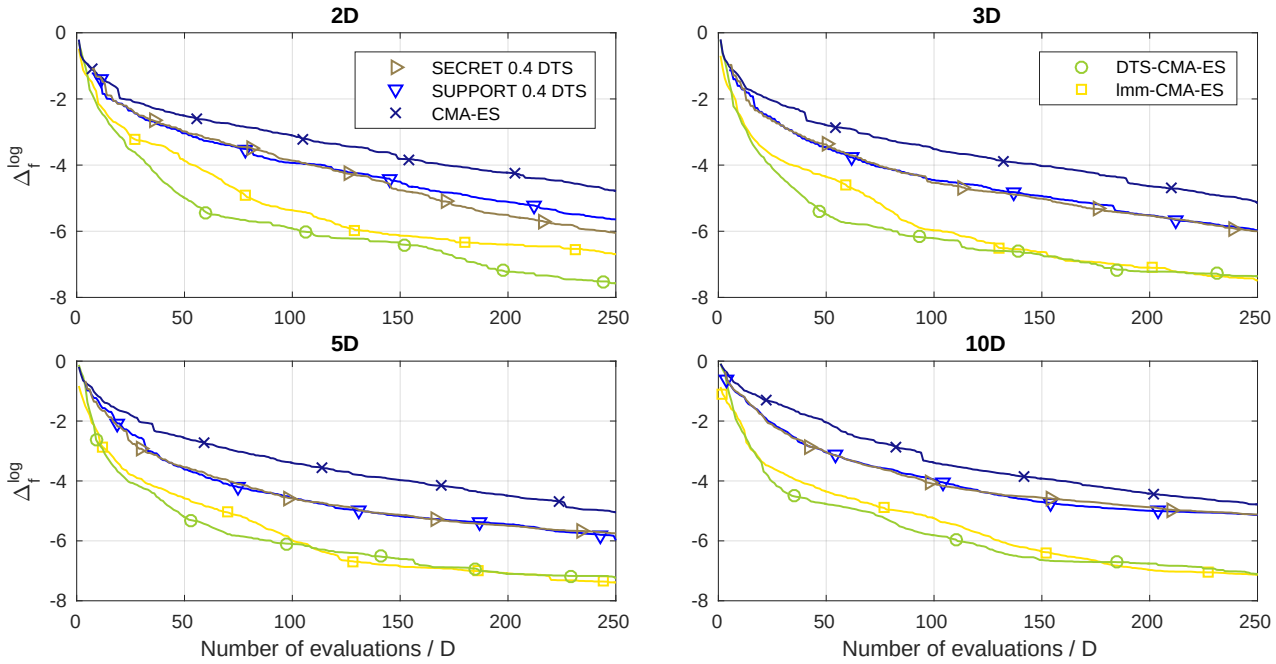

Figure 4: Scaled median distances $\Delta_f^{\log}$ averaged over all 24 COCO functions in $2D$, $3D$, $5D$, and $10D$ for algorithms CMA-ES, DTS-CMA-ES, lmm-CMA-ES, and 2 RF settings of DTS-CMA-ES.

sus metric Gaussian process regression in surrogate modelling for CMA evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 177–178, New York, NY, USA, 2017. ACM.

[20] Z. Pitra, L. Bajer, J. Repický, and M. Holeňa. Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1622–1629, New York, NY, USA, 2017. ACM.

[21] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. MIT Press, 2006.