Benchmarking GNN-CMA-ES on the BBOB noiseless testbed

Louis Faury
Criteo AI Lab
LTCI, Telecom ParisTech
Université Paris-Saclay, France
l.faury@criteo.com

Clément Calauzènes
Criteo AI Labs
32 Rue Blanche
Paris, France
c.calauzenes@criteo.com

Olivier Fercoq
LTCI, Telecom ParisTech
Université Paris-Saclay, France
olivier.fercoq@telecom-paristech.fr

ABSTRACT

We evaluate in this paper the GNN-CMA-ES algorithm on the BBOB noiseless testbed. The GNN-CMA-ES algorithm was recently proposed as a plug-in extension to CMA-ES, introducing the possibility to train flexible search distributions, in contrast to standard search distributions (such as the multivariate Gaussian). By comparing GNN-CMA-ES and CMA-ES, we show the benefits of this extension on some unimodal functions as well as on a variety of multimodal functions. We also identify a family of unimodal functions where GNN-CMA-ES can degrade the performances of CMA-ES and discuss the possible reasons behind this behavior.

CCS CONCEPTS

• **Computing methodologies** → **Continuous space search**;

KEYWORDS

Benchmarking, Black-box optimization, Evolutionary Strategies, Generative Neural Networks

ACM Reference format:
Louis Faury, Clément Calauzènes, and Olivier Fercoq. 2019. Benchmarking GNN-CMA-ES on the BBOB noiseless testbed. In *Proceedings of Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, July 13–17, 2019 (GECCO '19 Companion)*, 9 pages.
https://doi.org/10.1145/3319619.3326856

1 INTRODUCTION

The Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) [8, 14] is a stochastic method for continuous zeroth-order optimization of arbitrary (*e.g* non-convex, non-smooth) functions, and belongs to the general class of Evolutionary Strategies (ES) [19, 22]. In a few words, CMA-ES maintains a multivariate normal search distribution which it updates via heuristic mechanisms such as covariance matrix adaptation. This allows the efficient optimization of a variety of *hard* (ill-conditioned, non-separable) unimodal functions and also enables the search distribution to potentially escape local minima of multimodal functions via an efficient implicit exploration/exploitation trade-off.

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326856

In recent work [5], we argued that the rigidity of the normal distribution could potentially be harmful to the stochastic search. For instance, its lack of flexibility could lead an algorithm like CMA-ES to reduce the entropy of the search distribution so that it *locally* fits the objective function level curves. Such a shortage in entropy will impact the quality of the exploration phase, by either slowing down the discovery of the minimum (for instance in a curved valley), or taking away some probability mass on a (local) minimum, which could be revealed to be the global minima with more function evaluations.

In order to circumvent such short-comings and maintain *flexible* search distributions that could adapt to a variety of objective functions, we proposed in [5] to use *bijective* Generative Neural Networks in ES algorithms. Generative Neural Networks (GNNs) [17] have been studied in the context of *density estimation* and shown to be able to model complex and multimodal distributions. While in all generality they have intractable densities (justifying their adversarial training [7]), recent work [3, 4, 20] focused on building bijective GNNs (also known as normalizing flows), for which the density can be exactly computed. As shown in [5], this enables an efficient gradient-based training of GNNs for an ES purpose.

We focus on one particular bijective GNN: the NICE [3] model. It allows to model highly flexible distributions at minimal cost, and induces a *volume preserving* transformation. This means that the volume of the distribution is fully determined by the *latent* distribution of the GNN (typically a multivariate normal). The exploration/exploitation trade-off is therefore left entirely to the latent space, which can be shaped by literally any ES algorithm operating on the latent distribution.

In this paper, we base on [5] and focus on the case where the GNN's latent distribution is a multivariate normal, trained by CMA-ES. We denote this algorithm GNN-CMA-ES and detail its procedure in Section 2. We detail the experimental procedure we followed in Section 3 and provide experimental results on the BBOB testbed in Section 4. Finally, we discuss these results in Section 5.

2 GNN-CMA-ES

2.1 Notation

In the following, we note f the objective to be minimized and $x \in \mathcal{X}$ its D-dimensional argument. Samples x from the GNN are obtained by applying a bijective mapping g_η to samples $z \in \mathcal{Z}$ drawn from a latent distribution ν_ω. The symbol η represents the weight and biases of the neural networks implemented by the NICE model to build g_η, which structure can be found in [5] or in the original NICE paper [3]. The latent distribution ν_ω is set to be a multivariate normal with mean μ and covariance matrix Σ, described jointly by the parameter $\omega \triangleq \langle \mu, \Sigma \rangle$. Finally, we note $\pi_{\omega,\eta}$

the distribution generated by the GNN and will use the notation θ to jointly describe the parameters ω and η - that is $\theta \triangleq (\omega, \eta)$.

2.2 Principle

We train GNN distributions with an ES perspective by minimizing the objective:

$$J(\theta) = J(\omega, \eta) \triangleq \mathbb{E}_{x\sim\pi_{\omega,\eta}}\left[f(x)\right] \tag{1}$$

which gradients w.r.t θ can be obtained thanks to the log-trick and estimated from Monte-Carlo sampling [25]. This gradient evaluation only requires being able to differentiate the log-density of $\pi_{\omega,\eta}$. Thanks to the bijectivity of g_η, this density can easily be evaluated (and differentiated) thanks to the change of variable formula. Indeed, noting $h_\eta \triangleq g_\eta^{-1}$ we have:

$$\pi_{\omega,\mu}(x) = \nu_\omega(h_\eta(x))\left|\frac{\partial h_\eta(x)}{\partial x}\right| \tag{2}$$

The NICE model maintains a unit Jacobian ($|\partial h_\eta/\partial x| = 1$) and allows to compute h_η for the same price as a feed-forward pass on g_η. This allows to efficiently access stochastic gradients of (1) and train GNNs for an ES context in a principled manner.

Unfortunately, jointly training ω and η via the objective (1) ultimately fails, even with second-order approach like natural gradient descent (leading to instances of the Natural Evolutionary Strategies [24]). In [5], we derive an efficient training technique that performs an *alternated minimization* of $J(\omega, \eta)$. More precisely, at every generation t of the ES procedure we successively (approximately) solve:

$$\omega_{t+1} = \underset{\omega}{\operatorname{argmin}}\, J(\omega, \eta_t) \tag{3a}$$

$$\eta_{t+1} = \underset{\eta}{\operatorname{argmin}}\, J(\omega_{t+1}, \eta) \tag{3b}$$

The optimization of the latent distribution parameters (3a) is left to CMA-ES, which operates on the multivariate normal distribution ν_ω. Note that by rewriting the objective (3a):

$$J(\omega, \eta_t) = \mathbb{E}_{z\sim\nu_\omega}\left[f(g_{\eta_t}(z))\right] \tag{4}$$

this procedure is equivalent to running CMA-ES on the (non-stationary) objective function $f \circ g_{\eta_t}$.

The optimization of the parameters η (ruling the flexibility of the distribution $\pi_{\omega,\eta}$) in (3b) is done by solving a trust-region problem, augmented with *inverse propensity scores* to take into account the change in distribution brought by (3a) and unbias the empirical estimation of $\mathbb{E}_{\pi_{\omega_{t+1},\eta}}\left[f(x)\right]$ with samples from π_{ω_t,η_t}. Concretely, we obtain η_{t+1} by (approximately) solving the program:

$$\begin{aligned} &\min_{\eta} \mathbb{E}_{x\sim\pi_{\omega_t,\eta_t}}\left[\frac{\pi_{\omega_{t+1},\eta}(x)}{\pi_{\omega_t,\eta_t}(x)} f(x)\right] \\ &\text{s.t } \mathrm{KL}\left(\pi_{\omega_{t+1},\eta}||\pi_{\omega_{t+1},\eta_t}\right) \leq \varepsilon \end{aligned} \tag{5}$$

where ε is a hyper-parameter given by the user that controls the change in distribution the GNN optimization is allowed to bring. The Kullback-Leibler (KL) divergence is approximated from samples, as well as the expectation under π_{ω_t,η_t}. The constraint is enforced via an adaptive penalization scheme (see [21]), and an optimal point for the penalized objective can be readily (approximately) discovered by a stochastic gradient descent algorithm. The role of the trust-region is to avoid degeneracies that are known to happen when minimizing inverse propensity scores [23] and to prevent the GNN from overfitting.

The objective in the program (5) is known as *off-policy*, meaning that it incorporates past samples in its optimization loop without adding bias to the estimation. It can therefore be readily augmented with samples from past generations, that is from $\pi_{\omega_{t-1},\eta_{t-1}}, \dots, \pi_{\omega_{t-T},\eta_{t-T}}$ (noting T the history's horizon). This technique can be useful when the population size λ of the ES algorithm is small, in order to ensure sufficient data exposure for the GNN. This setting was not evaluated in [5] and is leveraged here to be able to use CMA-ES with its default population size (rather small compared to usual batch sizes used to train GNNs).

The GNN-CMA-ES algorithm can be understood as a plug-in extension to CMA-ES, enabling to learn diverse and flexible search distributions which adapt their density level curves with those of the objective function. The CMA-ES algorithm operates directly on the composed function $f \circ g_\eta$, where g_η is learned by (3a). This allows the representation of the objective function in the latent space to be better suited for the stochastic search with a normal distribution, potentially leading to faster discovery of the global minimum. The exploration/exploitation trade-off is left to CMA-ES, which fully rules the volume of the search distribution.

2.3 Algorithm

We provide in Algorithm 1 the pseudo-code for GNN-CMA-ES. This generic algorithm can readily incorporate extension to CMA-ES, like restart and increasing population sizes (the restart being dictated only by the CMA-ES algorithm). We indicate in Algorithm 1 only the hyper-parameters linked with the GNN extension, and use for the remaining the CMA-ES default configuration. These hyper-parameters include the KL radius ε, the sample size M used to estimate the KL divergence in (5), the initial value of the Lagrangian multiplier β_0 used to solve (5), the horizon T and the structure of the Multi-Layer Perceptrons (MLPs) used in the NICE model.

We implement the NICE model using the Tensorflow Probabilities [2] (TfP) library. Relying on Tensorflow automatic differentiation, the penalized version of (5) is optimized with Adam [16] until a minimum is discovered.

3 EXPERIMENTAL PROCEDURE

We use the IPOP-CMA-ES [1] implementation of the PyCMA [9] library (version 2.6.0), both as baseline and as inner optimizer of GNN-CMA-ES. We therefore benchmark GNN-IPOP-CMA-ES in this paper, which we keep short to GNN-CMA-ES to reduce clutter. For all hyper-parameters relative to IPOP-CMA-ES, we use their default (adapted) value from PyCMA (both for the baseline and the inner optimization on GNN-CMA-ES). Similarly, we use in all our experiments the same hyper-parameters for GNN-CMA-ES, which we provide in Table 1. The NICE model is also kept constant in all experiments, and set to have three coupling layers with a one hidden layer MLP with 16 units and hyperbolic tangent activation. The computational overhead induced by the GNN is therefore quite modest (at least for GNNs standards), however enough to bring significant flexibility - as demonstrated in [5].

Algorithm 1: GNN-CMA-ES algorithm (with historic data)

inputs : Objective function f, initial mean μ_0, initial scale σ_0

hyper-parameters : KL radius ε, KL batch size M, NICE model architecture, initial weights η_0, initial penalty β_0, horizon T

(Initialization)
  Initialize CMA-ES with mean μ_0 and variance $\sigma_0 \cdot \mathbf{I}_D$
  Initialize NICE MLPs weights and biases with η_0.
  Let $\lambda \leftarrow 4 + \lfloor 3 \log D \rfloor$
  Let $\mathcal{H}$ be a circular buffer of length $T \times \lambda$
while *not terminate* **do**
  (Sampling)
    Sample $Z = \{z_1, \dots z_\lambda\} \overset{\text{i.i.d}}{\sim} \nu_{\omega_t}$
    Apply g_{η_t} to Z, obtain $X = \{x_1, \dots x_\lambda\} \overset{\text{i.i.d}}{\sim} \pi_{\omega_t, \eta_t}$
    Evaluate $L = \left\{\pi_{\omega_t, \eta_t}(x_1), \dots \pi_{\omega_t, \eta_t}(x_\lambda)\right\}$
    Evaluate $F = \{f(x_1), \dots, f(x_\lambda)\}$.
    Let $\mathcal{H} \leftarrow \mathcal{H} + \{X, L, F\}$
  (CMA-ES iteration)
    Apply CMA-ES to ν_{ω_t} with samples Z and evaluations F.
    Obtain ω_{t+1}
  (NICE iteration)
    `//KL divergence estimation`
    Sample $\tilde{x}_1, \dots, \tilde{x}_M \overset{\text{i.i.d}}{\sim} \pi_{\omega_{t+1}, \eta_t}$.
    $\tilde{KL}(\omega_{t+1}, \eta) \leftarrow \frac{1}{M} \sum_{i=1}^{M} \log\left(\frac{\pi_{\omega_{t+1}, \eta_t}(\tilde{x}_i)}{\pi_{\omega_{t+1}, \eta}(\tilde{x}_i)}\right)$
    `//Gradient based optimization`
    $\eta_{t+1} \in$
      $\operatorname{argmin}_\eta \left\{\frac{1}{T \cdot \lambda} \sum_{x, \pi_0, f \in \mathcal{H}} f \frac{\pi_{\omega_{t+1}, \eta}(x)}{\pi_0} + \beta_t \tilde{KL}(\omega_{t+1}, \eta)\right\}$
  (Penalty update)
    if $\tilde{KL}(\omega_{t+1}, \eta_{t+1}) > 2\varepsilon$ **then**
      $\beta_{t+1} \leftarrow 1.5\beta_t$
    end
    else if $\tilde{KL}(\omega_{t+1}, \eta_{t+1}) < \varepsilon/2$ **then**
      $\beta_{t+1} \leftarrow \beta_t / 1.5$
    end
end

We run IPOP-CMA-ES and GNN-CMA-ES both with a budget of $10^4 \times D$ on the BBOB 2018 noiseless function suites in four different dimensions (D=2,3,5 and 10). We follow the default evaluation procedure presented in the demonstration files of the Comparing Continuous Optimizers (COCO) code base (version 2.2.2). For each run of the algorithms, an initial starting point x_0 is sampled uniformly in $[-5, 5]^D$ and the initial step-size is set to $\sigma = 2.0$. For both algorithms, the maximum number of restarts is set to 6.

4 RESULTS

Results from experiments according to [15] and [10] on the benchmark functions given in [6, 13] are presented in Figures 1, 2 and 3 and in Tables 2 and 3. The experiments were performed with COCO [12] version 2.2.2, the plots produced with version 2.2.2.

Table 1: GNN-CMA-ES default hyper-parameters

Hyper-parameter	Symbol	Value
MLPs initialization	η_0	Glorot initialization
Initial KL regularization	β_0	1.0
KL radius	ε	0.01
Horizon	T	3
KL sample size	M	$100 \times D$

The **average runtime (aRT)**, used in the figures and tables, depends on a given target function value, $f_t = f_{\text{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and divided by the number of trials that actually reached f_t [11, 18]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

5 DISCUSSION

The relative performance of GNN-CMA-ES with respect to IPOP-CMA-ES greatly varies among the different objective functions of the BBOB testbed. As shown in Figures 1,2 and 3, GNN-CMA-ES accelerates CMA-ES on some (unimodal and multimodal) functions. Such acceleration is expected given the main motivation behind GNN-CMA-ES, which is flexibility of the search distribution. With additional flexibility, the search distribution can continuously adapt its search directions (useful on f_8 and f_9 for instance) and concurrently keep track of several local minima. On the other hand, GNN-CMA-ES degrades CMA-ES performances on some separable unimodal and ill-conditionned functions (f_2 and f_{11} for instance). While this is not satisfying, it is quite natural that the multivariate normal performs better on most of these functions. Indeed, we believe that since most of them have ellipsoidal shapes, the *rigidity* of the Gaussian serves as an *implicit prior* and the search distribution can efficiently align its density level lines with those of the objective functions. One can also witness that on the unimodal functions where GNN-CMA-ES accelerates CMA-ES, this improvement in performances seems to vanish as the dimension increases.

These experimental results therefore validate the benefits of using flexible search distributions in ES algorithms on *some* objective functions. They also highlight the fact that on relatively *simple* objectives, the normal search distribution is perfectly adapted and the additional flexibility brought by GNN-CMA-ES can degrade the convergence rate of CMA-ES.

As a consequence, future work could therefore focus on reducing the impact of GNN-CMA-ES on such functions while still allowing acceleration on non (locally) separable and multimodal functions with weak structure. We also plan on investigating the sensitivity of GNN-CMA-ES with respect to its hyper-parameters (such as ε and T,

that are here kept constant in all experiments) to design dimension-adapted optimal values. Finally, since the GNN extension proposed in [5] holds for virtually *any* ES algorithms, further benchmarking could be performed with other ES algorithms, such as the Natural Evolutionary Strategies family.

REFERENCES

[1] Anne Auger and Nikolaus Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, Vol. 2. IEEE, 1769–1776.

[2] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. 2017. Tensorflow distributions. *arXiv preprint arXiv:1711.10604* (2017).

[3] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. NICE: Non-Linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516* (2014).

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density Estimation using Real NVP. *arXiv preprint arXiv:1605.08803* (2016).

[5] Louis Faury, Clement Calauzenes, Olivier Fercoq, and Syrine Krichen. 2019. Improving Evolutionary Strategies with Generative Neural Networks. *arXiv preprint arXiv:1901.11271* (2019).

[6] S. Finck, N. Hansen, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions.* Technical Report 2009/20. Research Center PPE. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

[8] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[9] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. 2019. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634. (Feb. 2019). https://doi.org/10.5281/zenodo.2559634

[10] N. Hansen, A Auger, D. Brockhoff, D. Tušar, and T. Tušar. 2016. COCO: Performance Assessment. *ArXiv e-prints* arXiv:1605.03560 (2016).

[11] N. Hansen, A. Auger, S. Finck, and R. Ros. 2012. *Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup.* Technical Report. INRIA. http://coco.gforge.inria.fr/bbob2012-downloads

[12] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016).

[13] N. Hansen, S. Finck, R. Ros, and A. Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Technical Report RR-6829. INRIA. http://coco.lri.fr/downloads/download15.03/bbobdocfunctions.pdf Updated February 2010.

[14] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195.

[15] N. Hansen, T. Tušar, O. Mersmann, A. Auger, and D. Brockhoff. 2016. COCO: The Experimental Procedure. *ArXiv e-prints* arXiv:1603.08776 (2016).

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] David JC MacKay. 1995. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 354, 1 (1995), 73–80.

[18] Kenneth Price. 1997. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*. 153–157.

[19] Ingo Rechenberg. 1978. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*. Springer, 83–114.

[20] Oren Rippel and Ryan Prescott Adams. 2013. High-dimensional Probability Estimation with Deep Density Models. *arXiv preprint arXiv:1302.5125* (2013).

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[22] Hans-Paul Schwefel. 1977. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie.* Birkhäuser.

[23] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *International Conference on Machine Learning*. 814–823.

[24] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. 2008. Natural Evolution Strategies. In *Evolutionary omputation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence)*. IEEE, 3381–3387.

[25] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist Reinforcement Learning. *Machine Learning* 8, 3-4 (1992), 229–256.

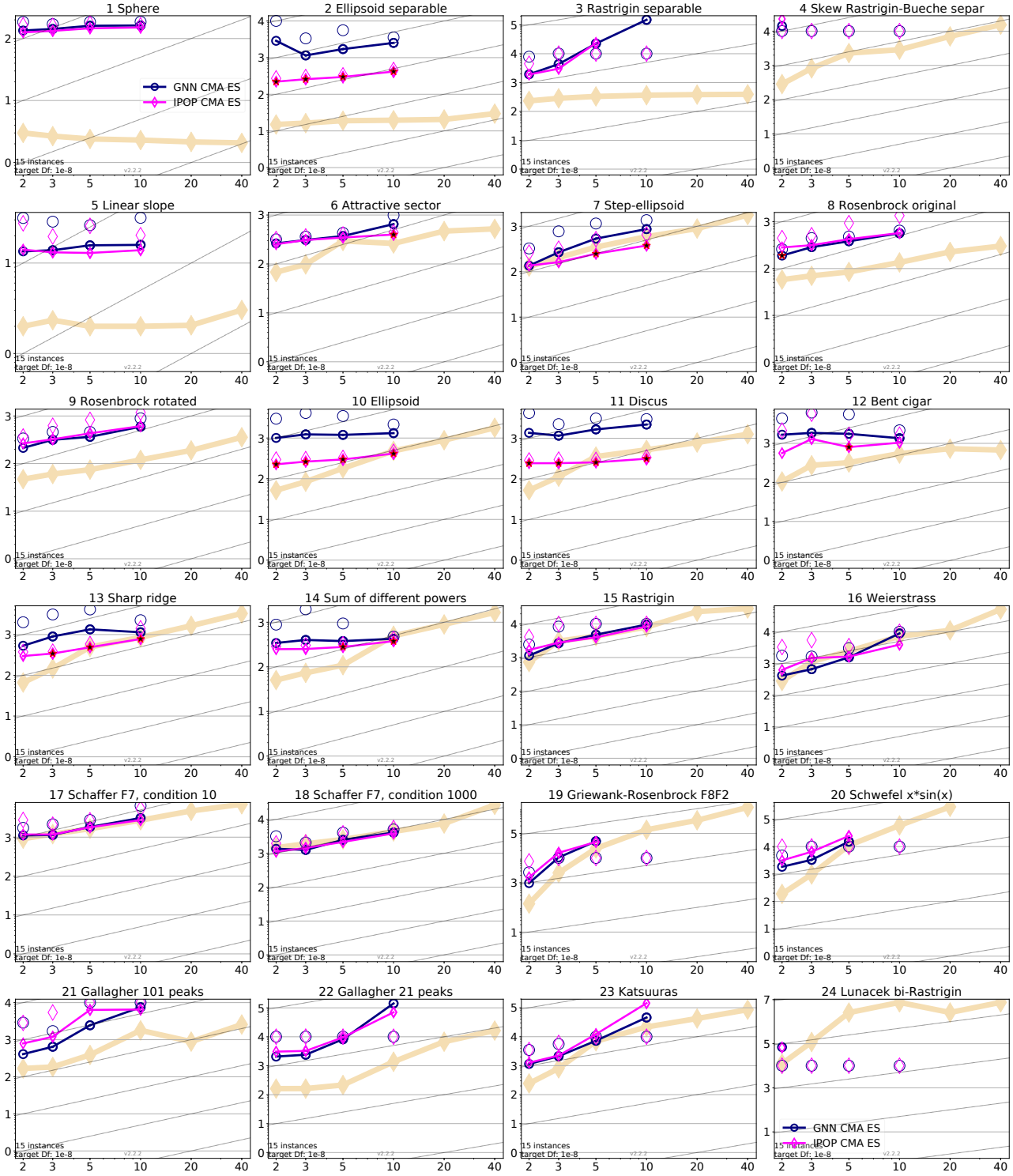

Figure 1: Average running time (aRT in number of f-evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: GNN-CMA-ES, ♢: IPOP-CMA-ES

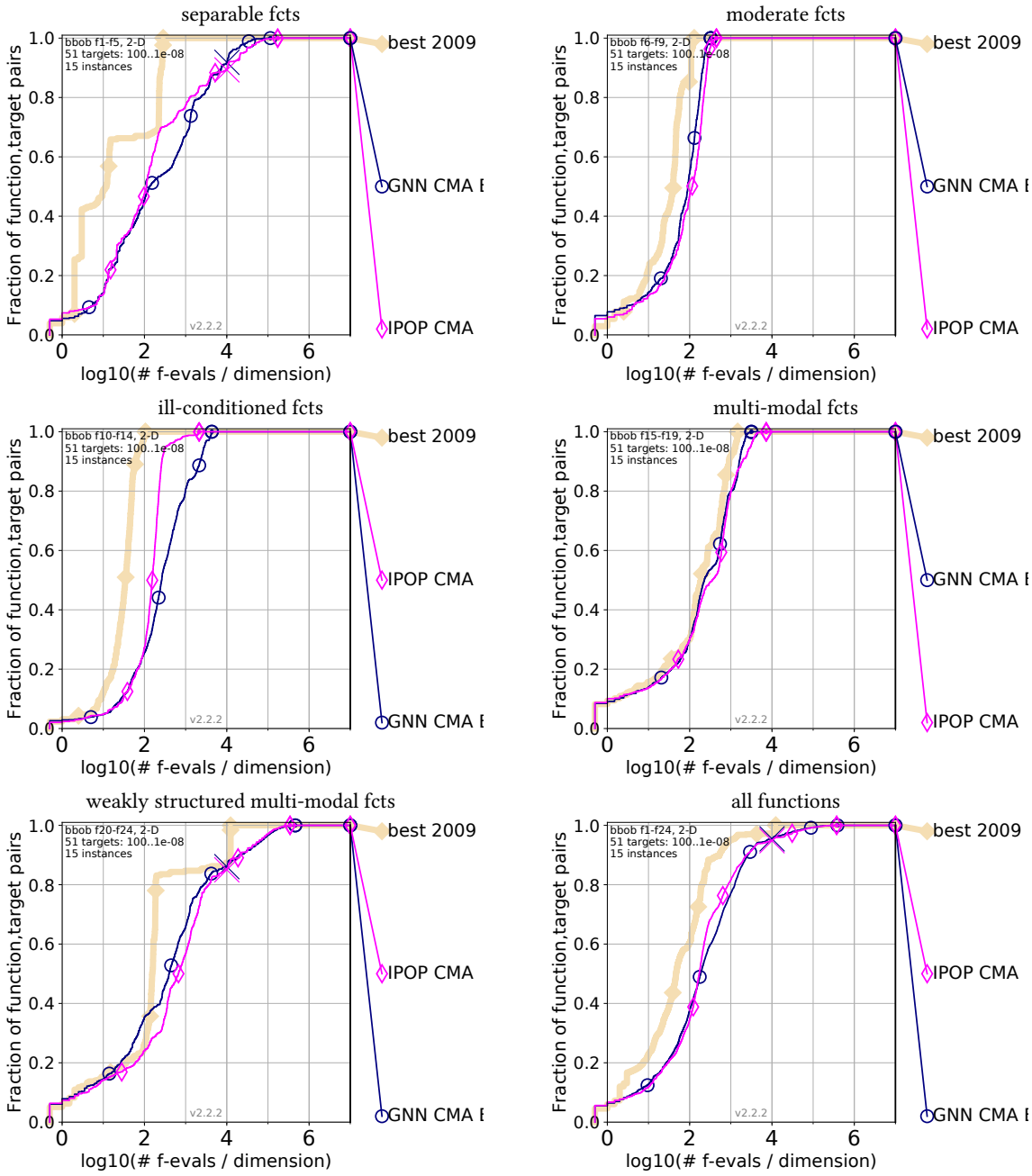

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 2-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

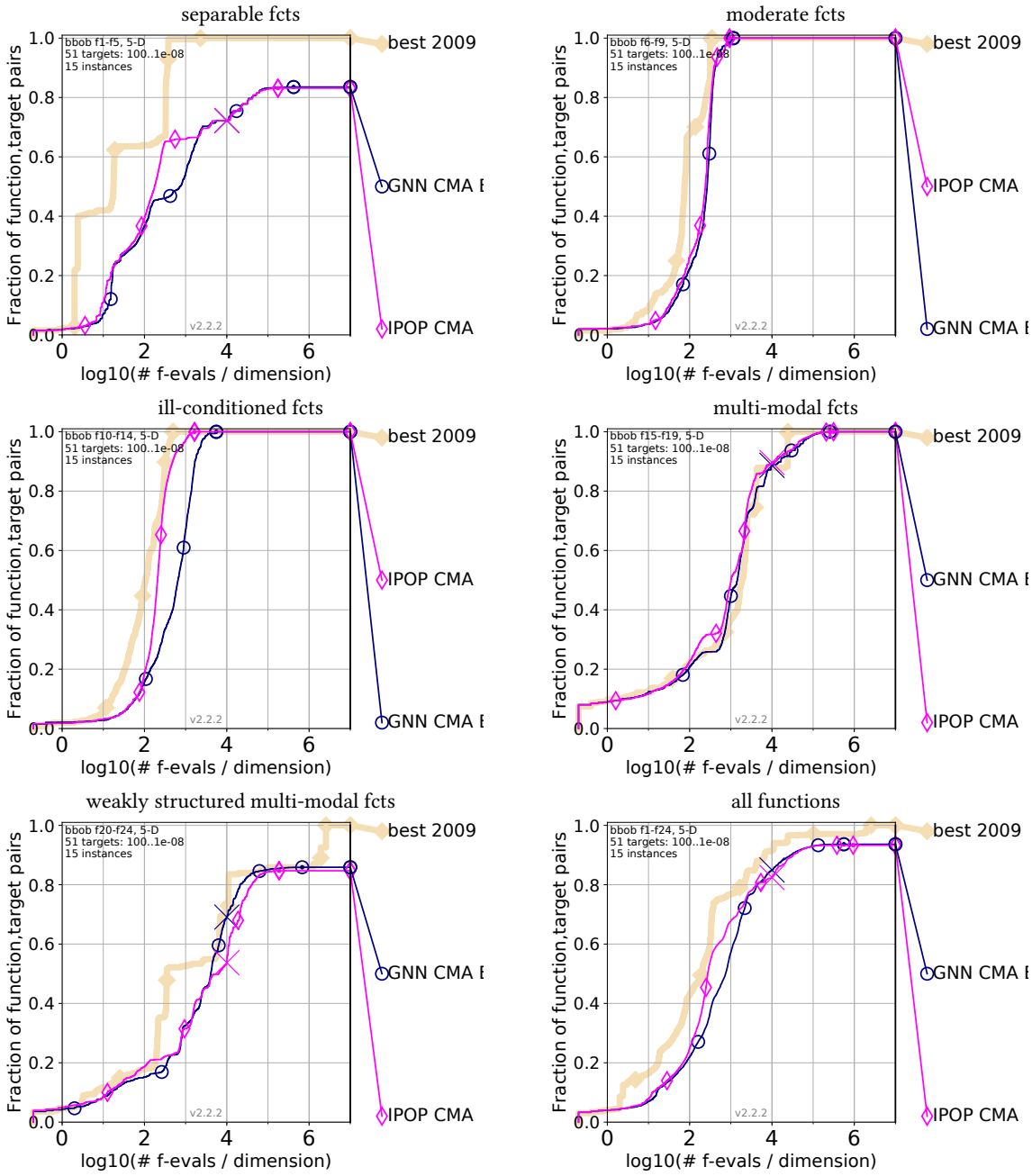

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 5-D. As reference algorithm, the best algorithm from BBOB 2009 is shown as light thick line with diamond markers.

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	2.0	6.0	6.0	6.0	6.0	6.0	6.0	15/15
GNN-CMA	4.0(5)	5.3(6)	10(7)	14(8)	20(4)	29(5)	39(4)	15/15
IPOP-CM	**2.8**(6)	**3.0**(4)	**8.0**(4)	**12**(5)	**16**(7)	**25**(6)	**37**(8)	15/15
f2	16	19	25	25	26	28	29	15/15
GNN-CMA	68(69)	145(27)	146(170)	167(160)	182(351)	188(280)	197(287)	14/15
IPOP-CM	**9.5**(5)	**11**(3)$^{\star 4}$	**10**(3)$^{\star 4}$	**11**(3)$^{\star 4}$	**11**(3)$^{\star 4}$	**13**(3)$^{\star 4}$	**14**(3)$^{\star 4}$	15/15
f3	15	271	445	446	450	454	464	15/15
GNN-CMA	**3.1**(2)	**2.3**(3)	7.3(12)	7.5(15)	7.7(6)	8.0(8)	8.2(15)	15/15
IPOP-CM	6.0(2)	2.9(5)	**6.6**(5)	**7.4**(8)	**7.6**(8)	**7.9**(9)	**8.1**(8)	15/15
f4	22	344	459	496	523	544	566	15/15
GNN-CMA	**1.9**(2)	**5.2**(6)	**37**(39)	**55**(54)	**52**(45)	**51**(59)	**50**(66)	7/15
IPOP-CM	1.9(1)	5.7(6)	95(66)	89(141)	85(105)	83(47)	80(115)	5/15
f5	4.0	4.0	4.0	4.0	4.0	4.0	4.0	15/15
GNN-CMA	4.1(4)	**6.2**(3)	**6.6**(6)	**6.7**(6)	**6.7**(6)	**6.7**(6)	**6.7**(6)	15/15
IPOP-CM	**3.1**(3)	6.5(2)	7.0(5)	7.0(6)	7.0(4)	7.0(3)	7.0(4)	15/15
f6	13	23	41	54	67	95	124	15/15
GNN-CMA	2.4(2)	**3.1**(2)	3.3(0.9)	3.5(0.8)	3.4(0.8)	3.5(0.8)	**3.6**(0.5)	15/15
IPOP-CM	**2.3**(3)	3.4(2)	**3.0**(0.5)	**3.1**(0.8)	**3.2**(0.8)	**3.5**(1)	3.7(0.6)	15/15
f7	3.0	21	60	193	217	217	241	15/15
GNN-CMA	4.6(8)	2.6(3)	**2.1**(1)	**0.80**(0.4)	**0.79**(0.5)	**0.79**(0.8)	0.92(0.7)	15/15
IPOP-CM	**3.6**(5)	**2.3**(3)	2.4(2)	0.82(0.5)	0.84(0.5)	0.84(0.5)	**0.92**(0.6)	15/15
f8	5.0	12	37	46	86	94	112	15/15
GNN-CMA	**3.3**(4)	**3.2**(4)	**2.8**(2)$^{\star 3}$	**4.0**(2)$^{\star 3}$	**2.6**(0.7)$^{\star 2}$	**3.0**(0.9)$^{\star 2}$	**3.1**(0.9)$^{\star 3}$	15/15
IPOP-CM	4.1(3)	7.0(5)	7.9(2)	8.0(3)	4.8(0.9)	5.0(1)	4.8(0.9)	15/15
f9	1	18	30	44	68	81	92	15/15
GNN-CMA	**1**(0)	**2.8**(2)	**4.3**(3)	**5.0**(3)	**4.0**(2)	**4.2**(2)	**4.3**(2)	15/15
IPOP-CM	**1**(0)	3.5(3)	7.7(5)	7.5(4)	5.4(2)	5.5(1)	5.5(2)	15/15
f10	30	46	54	61	68	82	98	15/15
GNN-CMA	21(51)	28(23)	29(51)	29(36)	27(35)	23(23)	20(26)	15/15
IPOP-CM	**4.4**(3)	**4.1**(2)	**4.6**(1)*	**4.4**(1)$^{\star 2}$	**4.4**(2)$^{\star 2}$	**4.5**(0.4)*	**4.3**(0.6)*	15/15
f11	35	45	50	62	67	81	97	15/15
GNN-CMA	17(33)	34(46)	42(58)	37(12)	38(52)	32(44)	28(34)	15/15
IPOP-CM	**5.5**(2)	**5.2**(2)*	**5.6**(1)$^{\star 2}$	**4.9**(0.7)$^{\star 3}$	**4.8**(1)$^{\star 3}$	**4.8**(0.8)$^{\star 3}$	**4.7**(0.7)$^{\star 3}$	15/15
f12	35	46	75	94	105	153	195	15/15
GNN-CMA	26(38)	39(47)	33(43)	29(42)	27(37)	21(19)	17(13)	15/15
IPOP-CM	**5.9**(10)	**8.1**(2)*	**6.5**(13)*	**6.5**(3)$^{\star 2}$	**6.6**(18)$^{\star 2}$	**5.9**(12)*	**5.4**(2)*	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	23	35	46	60	71	95	122	15/15
GNN-CMA	**2.1**(1)	**3.1**(2)	5.4(0.9)	7.9(1)	7.7(13)	7.5(18)	7.8(9)	15/15
IPOP-CM	3.2(2)	4.3(2)	**4.5**(1)	**4.3**(1.0)	**4.3**(1)	**4.4**(0.8)	**4.4**(0.7)	15/15
f14	1	7.0	16	24	38	67	90	15/15
GNN-CMA	**3.5**(3)	3.7(3)	4.3(2)	4.3(2)	**4.5**(0.9)	4.8(0.9)	5.6(0.9)	15/15
IPOP-CM	4.3(7)	**3.2**(4)	**3.6**(0.9)	**4.1**(1)	4.7(1)	**4.8**(1)	**4.7**(1)	15/15
f15	37	291	1033	1066	1113	1231	1412	15/15
GNN-CMA	2.2(1.0)	**2.5**(2)	**1.8**(1)	**1.8**(2)	**1.8**(1)	**1.7**(1)	**1.6**(1)	15/15
IPOP-CM	**1.3**(0.7)	2.7(2)	2.6(3)	2.8(3)	2.7(3)	2.6(3)	2.3(2)	15/15
f16	9.0	50	174	326	358	409	538	15/15
GNN-CMA	3.7(4)	**6.3**(13)	**2.7**(5)	**1.6**(3)	**1.6**(0.2)	**1.8**(2)	**1.5**(1)	15/15
IPOP-CM	**3.4**(1)	9.1(9)	5.1(3)	2.9(1)	2.7(5)	2.8(5)	2.3(3)	15/15
f17	3.0	61	133	275	396	1086	1657	15/15
GNN-CMA	16(2)	**1.5**(5)	1.3(1)	**1.1**(0.1)	1.6(1)	**1.4**(1)	**1.3**(0.7)	15/15
IPOP-CM	**1.7**(2)	1.6(0.7)	**1.2**(2)	1.5(0.9)	**1.3**(2)	1.6(1)	1.4(1)	15/15
f18	19	134	666	1249	1708	2438	2858	15/15
GNN-CMA	**4.1**(3)	3.2(2)	0.89(0.6)	**0.61**(0.5)	0.70(0.7)	0.96(0.4)	0.91(0.6)	15/15
IPOP-CM	5.1(14)	**3.0**(4)	**0.83**(0.8)	0.67(0.8)	**0.56**(0.2)	**0.65**(0.4)	**0.75**(0.5)	15/15
f19	1	1	26	216	227	252	276	15/15
GNN-CMA	**1**(0)	**1**(0)	11(3)	**7.4**(7)	**7.4**(10)	**7.2**(7)	**6.8**(6)	15/15
IPOP-CM	**1**(0)	**1**(0)	**6.4**(8)	8.3(5)	10(14)	11(11)	12(10)	15/15
f20	4.0	61	365	366	366	370	375	15/15
GNN-CMA	4.0(4)	**5.4**(4)	**7.0**(6)	**8.4**(9)	**8.9**(9)	**9.3**(8)	**10**(7)	15/15
IPOP-CM	**2.8**(2)	9.0(10)	11(16)	15(6)	16(15)	16(15)	16(19)	14/15
f21	2.0	51	174	276	290	324	330	15/15
GNN-CMA	**1.1**(0.5)	**3.3**(9)	**2.8**(7)*	**2.3**(0.5)*	**2.2**(0.5)*	**2.3**(7)*	**2.4**(7)	15/15
IPOP-CM	1.3(0.5)	5.3(5)	7.3(12)	4.7(6)	4.8(8)	4.5(6)	4.7(5)	15/15
f22	5.0	27	168	218	249	289	306	15/15
GNN-CMA	1.1(1)	**10**(10)	**4.7**(4)	**4.7**(9)	**9.2**(1)	**14**(38)	**13**(19)	13/15
IPOP-CM	**1.0**(2)	10(12)	12(21)	13(16)	15(24)	21(25)	20(32)	13/15
f23	8.0	193	234	263	299	348	379	15/15
GNN-CMA	2.6(3)	**4.2**(3)	**6.7**(3)	**7.0**(5)	**6.4**(4)	**6.0**(6)	**5.9**(6)	15/15
IPOP-CM	**2.2**(1)	4.6(4)	7.7(3)	7.2(5)	7.3(6)	6.7(5)	6.5(6)	15/15
f24	18	857	8515	23399	24113	24721	24721	5/15
GNN-CMA	**1.5**(0.9)	**8.0**(7)	5.5(5)	5.9(6)	5.7(4)	5.6(12)	5.6(7)	2/15
IPOP-CM	1.9(1)	18(36)	**5.1**(5)	**5.7**(5)	**5.5**(7)	**5.4**(5)	**5.4**(4)	2/15

Table 2: Average runtime (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2009 in dimension 2. This aRT ratio and, in braces as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding reference aRT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of functions (24). A $\downarrow$ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.2.2

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	12	15/15
GNN-CMA	**3.0**(1)	12(4)	19(5)	27(8)	34(4)	47(4)	60(7)	15/15
IPOP-CM	3.2(2)	**10**(3)	**17**(5)	**23**(4)	**29**(7)	**41**(6)	**54**(6)	15/15
f2	83	87	88	89	90	92	94	15/15
GNN-CMA	35(22)	58(32)	80(73)	86(123)	89(79)	90(41)	90(86)	15/15
IPOP-CM	**8.3**(2)$^{\star 3}$	**10**(1)$^{\star 4}$	**11**(0.6)$^{\star 4}$	**11**(1)$^{\star 4}$	**12**(1)$^{\star 4}$	**14**(1)$^{\star 4}$	**15**(1)$^{\star 4}$	15/15
f3	716	1622	1637	1642	1646	1650	1654	15/15
GNN-CMA	2.2(2)	16(18)	**67**(56)	**67**(82)	**67**(71)	**68**(48)	**68**(54)	5/15
IPOP-CM	**0.73**(0.8)	**8.7**(16)	70(60)	70(93)	70(183)	70(76)	71(71)	5/15
f4	809	1633	1688	1758	1817	1886	1903	15/15
GNN-CMA	2.2(3)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
IPOP-CM	**2.1**(3)	∞	∞	∞	∞	∞	∞ *5e4*	0/15
f5	10	10	10	10	10	10	10	15/15
GNN-CMA	5.5(0.9)	7.7(3)	7.9(3)	7.9(1)	7.9(2)	7.9(4)	7.9(2)	15/15
IPOP-CM	**4.5**(3)	**6.1**(2)	**6.4**(2)	**6.5**(3)	**6.5**(3)	**6.5**(3)	**6.5**(2)	15/15
f6	114	214	281	404	580	1038	1332	15/15
GNN-CMA	2.4(0.3)	2.0(0.1)	2.3(0.5)	2.1(0.3)	1.7(0.4)	1.3(0.1)	1.3(0.2)	15/15
IPOP-CM	**2.1**(0.9)	**1.8**(0.5)	**2.0**(0.4)	**1.7**(0.3)	**1.5**(0.2)	**1.2**(0.1)	**1.2**(0.1)	15/15
f7	24	324	1171	1451	1572	1572	1597	15/15
GNN-CMA	4.1(1.0)	2.0(2)	1.4(0.8)	1.4(0.7)	1.6(1)	1.6(1)	1.6(2)	15/15
IPOP-CM	**3.1**(2)	**1.1**(1)	**0.71**(0.4)	**0.66**(0.3)	**0.71**(0.5)	**0.71**(0.4)	**0.73**(0.5)	15/15
f8	73	273	336	372	391	410	422	15/15
GNN-CMA	3.4(0.9)	**2.9**(0.9)	**3.5**(0.5)	**3.7**(0.3)	**3.9**(0.4)	**4.1**(0.2)	**4.4**(0.8)	15/15
IPOP-CM	**2.9**(2)	3.9(3)	4.1(3)	4.3(0.5)	4.4(3)	4.6(2)	4.8(3)	15/15
f9	35	127	214	263	300	335	369	15/15
GNN-CMA	**5.4**(0.9)	**5.2**(2)	**5.2**(1)	**5.0**(1)	**4.8**(1)	**4.8**(1)	**4.7**(0.8)	15/15
IPOP-CM	6.0(2)	8.3(6)	6.8(5)	6.2(3)	5.8(5)	5.7(3)	5.7(2)	15/15
f10	349	500	574	607	626	829	880	15/15
GNN-CMA	5.8(5)	6.8(7)	8.0(8)	8.7(8)	8.8(9)	6.9(5)	6.7(5)	15/15
IPOP-CM	**2.0**(0.5)$^{\star 2}$	**1.7**(0.2)$^{\star 4}$	**1.7**(0.2)$^{\star 4}$	**1.7**(0.2)$^{\star 4}$	**1.8**(0.2)$^{\star 4}$	**1.5**(0.2)$^{\star 4}$	**1.6**(0.1)$^{\star 4}$	15/15
f11	143	202	763	977	1177	1467	1673	15/15
GNN-CMA	19(14)	25(8)	8.6(4)	7.3(3)	6.5(3)	5.4(3)	4.9(2)	15/15
IPOP-CM	**3.8**(0.5)$^{\star 2}$	**3.2**(0.7)$^{\star 4}$	**0.97**(0.2)$^{\star 4}$	**0.84**(0.1)$^{\star 4}$	**0.76**(0.0)$^{\star 4}$	**0.71**(0.1)$^{\star 4}$	**0.72**(0.1)$^{\star 4}$	15/15
f12	108	268	371	413	461	1303	1494	15/15
GNN-CMA	8.8(11)	11(11)	13(10)	14(9)	14(8)	6.1(2)	5.7(8)	15/15
IPOP-CM	**6.9**(5)	**4.8**(4)	**5.0**(3)$^{\star 2}$	**5.4**(4)$^{\star 2}$	**5.6**(4)$^{\star 2}$	**2.5**(2)$^{\star 2}$	**2.6**(1)$^{\star 2}$	15/15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f13	132	195	250	319	1310	1752	2255	15/15
GNN-CMA	3.1(0.4)	4.7(3)	6.3(2)	6.8(5)	2.3(2)	2.5(2)	2.7(0.7)	15/15
IPOP-CM	**2.6**(0.7)	**3.0**(0.8)	**3.2**(0.6)$^{\star 2}$	**3.3**(0.7)$^{\star 3}$	**0.92**(0.1)$^{\star 4}$	**1.0**(0.2)$^{\star 4}$	**1.00**(0.1)$^{\star 3}$	15/15
f14	10	41	58	90	139	251	476	15/15
GNN-CMA	2.5(4)	3.0(1)	3.8(0.4)	3.9(0.8)	3.9(0.6)	3.7(0.6)	3.0(2)	15/15
IPOP-CM	**1.4**(2)	**2.7**(0.5)	**3.7**(2)	**3.6**(0.7)	**3.6**(0.8)	**3.4**(0.7)	**2.6**(0.2)	15/15
f15	511	9310	19369	19743	20073	20769	21359	14/15
GNN-CMA	1.8(2)	**0.93**(1.0)	1.1(2)	1.1(0.4)	1.1(2)	1.1(0.8)	1.1(0.6)	14/15
IPOP-CM	**1.6**(2)	0.95(1.0)	**0.91**(0.3)	**0.91**(0.6)	**0.91**(0.9)	**0.91**(0.9)	**0.91**(0.5)	14/15
f16	120	612	2662	10163	10449	11644	12095	15/15
GNN-CMA	**2.8**(2)	**2.4**(2)	**1.6**(0.9)	**0.59**(0.2)	**0.67**(0.2)	**0.63**(0.2)	**0.64**(0.2)	15/15
IPOP-CM	3.0(2)	3.5(5)	1.8(1)	0.64(0.5)	0.70(0.5)	0.66(0.5)	0.67(0.5)	15/15
f17	5.0	215	899	2861	3669	6351	7934	15/15
GNN-CMA	**4.1**(5)	1.8(5)	1.2(2)	1.0(2)	1.1(0.4)	1.1(0.7)	1.1(0.4)	15/15
IPOP-CM	5.0(5)	**1.0**(0.5)	**0.89**(0.1)	**0.64**(2)	**0.56**(0.5)	**0.92**(0.3)	**1.1**(0.5)	15/15
f18	103	378	3968	8451	9280	10905	12469	15/15
GNN-CMA	1.3(0.7)	3.3(4)	0.84(0.4)	0.69(0.3)	0.75(0.7)	0.84(0.4)	0.97(0.6)	15/15
IPOP-CM	**0.91**(0.2)	**1.4**(0.3)	**0.56**(0.7)	**0.43**(0.4)	**0.45**(0.4)	**0.68**(0.4)	**0.83**(0.3)	15/15
f19	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
GNN-CMA	**1**(0)	**1**(0)	204(228)	2.3(3)	2.0(1)	2.0(2)	2.0(3)	3/15
IPOP-CM	**1**(0)	**1**(0)	**109**(73)	**1.6**(2)	**1.4**(1)	**1.4**(1)	**1.8**(2)	3/15
f20	16	851	38111	51362	54470	54861	55313	14/15
GNN-CMA	3.9(3)	**5.1**(7)	**1.7**(0.9)	**1.3**(1)	**1.2**(1)	**1.4**(0.9)	**1.4**(0.9)	8/15
IPOP-CM	**3.4**(2)	6.2(5)	3.1(3)	2.3(3)	2.2(1)	2.2(3)	2.2(3)	5/15
f21	41	1157	1674	1692	1705	1729	1757	14/15
GNN-CMA	**2.3**(1)	5.4(9)	**6.6**(3)	**6.7**(3)	**6.8**(4)	**6.8**(23)	**6.9**(16)	14/15
IPOP-CM	16(9)	**4.1**(8)	18(31)	18(12)	18(43)	18(50)	18(22)	10/15
f22	71	386	938	980	1008	1040	1068	14/15
GNN-CMA	10(16)	15(14)	**36**(34)	**40**(31)	**39**(50)	**39**(24)	**38**(76)	10/15
IPOP-CM	**4.6**(9)	**14**(102)	43(97)	41(67)	47(51)	46(39)	45(36)	8/15
f23	3.0	518	14249	27890	31654	33030	34256	15/15
GNN-CMA	4.4(4)	**6.9**(4)	**1.8**(2)	**1.00**(0.4)	**0.91**(1)	**1.1**(1)	**1.0**(1)	12/15
IPOP-CM	**3.3**(2)	11(7)	3.0(3)	1.9(2)	1.7(3)	1.6(2)	1.8(3)	8/15
f24	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
GNN-CMA	2.4(2)	**1.0**(0.7)	∞	∞	∞	∞	∞ *5e4*	0/15
IPOP-CM	**1.6**(1)	1.6(2)	∞	∞	∞	∞	∞ *5e4*	0/15

Table 3: Average runtime (aRT in number of function evaluations) divided by the respective best aRT measured during BBOB-2009 in dimension 5. This aRT ratio and, in braces as dispersion measure, the half difference between 10 and 90%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding reference aRT in the first row. The different target Δf-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of functions (24). A ↓ indicates the same tested against the best algorithm from BBOB 2009. Best results are printed in bold.

Data produced with COCO v2.2.2