Technical Report

Black Box Optimization Benchmarking

Amir Omeradzic

February 27, 2025

1 Introduction

Basic motivation:

- Improve global search of ES dynamically
- Vary/adapt population size by different means, i.e., by restarting or by online adaptation
- Implement ES applying principle of "mutate large and inherit small" [Bey98] (called "learning rate adaptation" in [NAO23])

Research goals:

- Re-implement state-of-the-art population control methods with CMA-ES
- Simplify algorithms to some extent, since some methods use ad-hoc and complex formulae for their adaptation
- Compare results with existing BBOB datasets (https://coco-platform.org/testsuites/bbob/data-archive.html):
  `CMAES-APOP_Nguyen, PSA-CMA-ES_Nishida, BIPOP-CMA-ES_Hansen_noiseless`
- First discussion of results
- Outlook to further possible research

Methods which are implemented and studied experimentally:

- Sec. 2.2: basic implementation of CMA-ES by Hansen
- Sec. 2.3: BIPOP-CMA-ES based on information from [Han09] and pseudocode of [BS17]
- Sec. 2.4: APOP-CMA-ES from [NH17] with simplifications from [OB24]
- Sec. 2.5: PSA-CMA-ES from [NA16, NA18a] with simplifications regarding normalization factors
- Sec. 2.6: LRA-CMA-ES from [NAO23] (and modified variant called ETA), to control the learning rate of the positional (and covariance) updates
- Sec. 2.7: first implementation of a Meta-CMSA-ES for population size control

2 Details on the Implementation

2.1 Experimental Setup

General remarks:

- Black-Box Optimization Benchmarking (BBOB, https://coco-platform.org/) tool was installed for Matlab
- Implementation and code for benchmarking is uploaded to https://github.com/omam-evo/coco
- Folder `bbob` contains data to execute BBOB benchmarking
  - `main_experiment.m` runs standard BBOB over all test functions
  - `main_coco_single.m` runs single trials for functions from BBOB test bed
  - experiment and postprocessing data of BBOB contains large files, which is stored locally by OMAM
- Folder `opt` contains the ES implementations
  - `main_dyn.m`: run dynamics on certain non-BBOB test functions (defined in `Fitness.m`)
  - `main_dyn.m` and `bbob\main_experiment.m` call the ES implementations from folder `opt`

2.2 Basic CMA-ES

Summary:

- path: `opt\purecmaes_basic.m`
- Basic CMA-ES implementation based on `purecmaes.m` in https://cma-es.github.io/cmaes_sourcecode_page.html
- the following simplifications to Hansen implementation were added: `hsig = 1`;
  condition `lambda > lambda/(c1+cmu)/N/10` for CMA update was removed.
- termination criteria were implemented using `opt\subroutines\terminate_cma.m`. Appropriate termination criteria improve performance on BBOB.
- `purecmaes_basic.m` is used in BIPOP-CMA-ES, and its code is also reused for the implementation of the subsequent CMA methods (except CMSA)

The termination criteria are chosen as

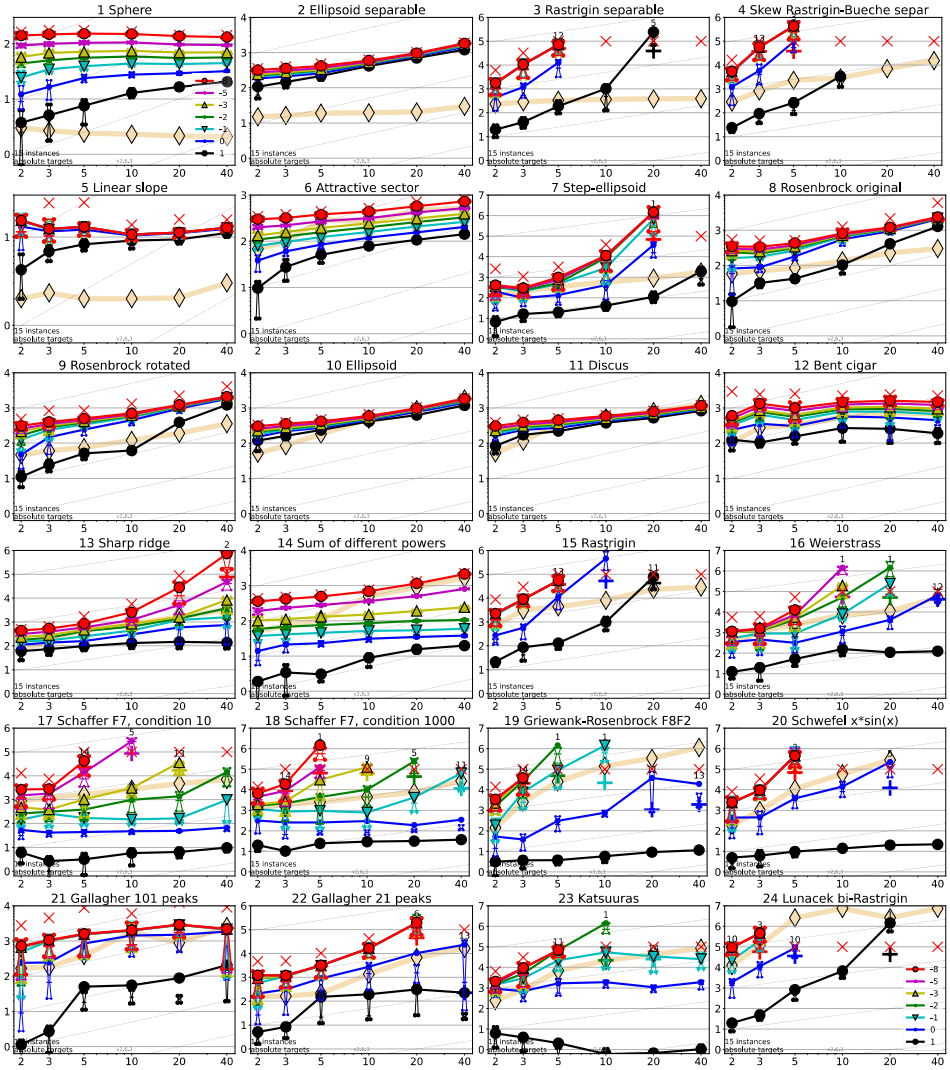
- function value stop: final BBOB final target was hit via `cocoProblemFinalTargetHit()` or own predefined f-stopping threshold was hit (non-BBOB run)
- maximum number of generations $g >$ 1e9 is exceeded (usually not needed for BBOB, since budget of function evaluations is the limiting factor)
- vanishing overall step-size $\sigma D_{\max} <$ 1e$-$12 ($D_{\max,\min}$: square root of maximum or minimum eigenvalue)
- square root of condition number exceeds threshold $D_{\max}/D_{\min} >$ 1e7
- vanishing progress in search space $\left\|\mathbf{y}^{(g+1)} - \mathbf{y}^{(g)}\right\| <$ 1e-12
- f-stagnation with $\max(\mathbf{f}_{\text{med}}) - \min(\mathbf{f}_{\text{med}}) <$ 1e-12, where the array $\mathbf{f}_{\text{med}}$ stores the median fitness values of all selected individuals over the last 10 generations, see also `opt\subroutines\stag_check.m`.

The termination criteria above are applied to all subsequent CMA-ES variants

The results in Fig. 1 can be used a reference for a simple CMA-ES with default settings and population size, since no simple CMA data was found in https://coco-platform.org/testsuites/bbob/data-archive.html.

Overview page

Scaling of run "time" with problem dimension



Scaling of runtime with dimension to reach certain target values Δf. Lines: expected runtime (ERT); Cross (+): median runtime of successful runs to reach the most difficult target that was reached at least once (but not always); Cross (×): maximum number of f-evaluations in any trial. Notched boxes: interquartile range with median of simulated runs; All values are divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown is the ERT for fixed target precision values of 10^k with k given in the legend. Numbers above ERT-symbols (if appearing) indicate the number of trials reaching the respective target. The light thick line with diamonds indicates the best algorithm from BBOB 2009 for the most difficult target. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

Figure 1: Basic CMA-ES (based on purecmaes.m by Hansen).

2.3 BIPOP

Summary:

- Path: `opt\bipop_cma_es.m`
- BIPOP was introduced in [Han09], but no pseudocode was provided. The ES is explained in text form. However, [BS17] provide pseudocode.
- `bipop_cma_es.m` uses a very similar code as [BS17], but with minor modifications:
  - maximum population size `lambda_max=2^9*lambda0` is enforced, after which no further restarts are conducted (see description in [Han09])
  - `lambda` is increased by `lambda = 2^n*lambda0` (without subtracting number of small population restarts, as done in [BS17])
  - small and large populations have same remaining budget of function evaluations (small population size has no additional generational or function evaluation limit, as described in [Han09, BS17])
- Here, we use the termination conditions from Sec. 2.2 of the simple CMA-ES.

The results in Fig. 2 show close agreement to the existing data provided by Hansen most of the time. Large differences only occur for f19, and especially for f24 (Lunacek bi-Rastirgin). The differences could be related to additional stopping criteria by Hansen and details of the BIPOP implementation, for which no pseudocode was given.

2.4 APOP

Simplified CMAES-APOP from [NH17] was implemented:

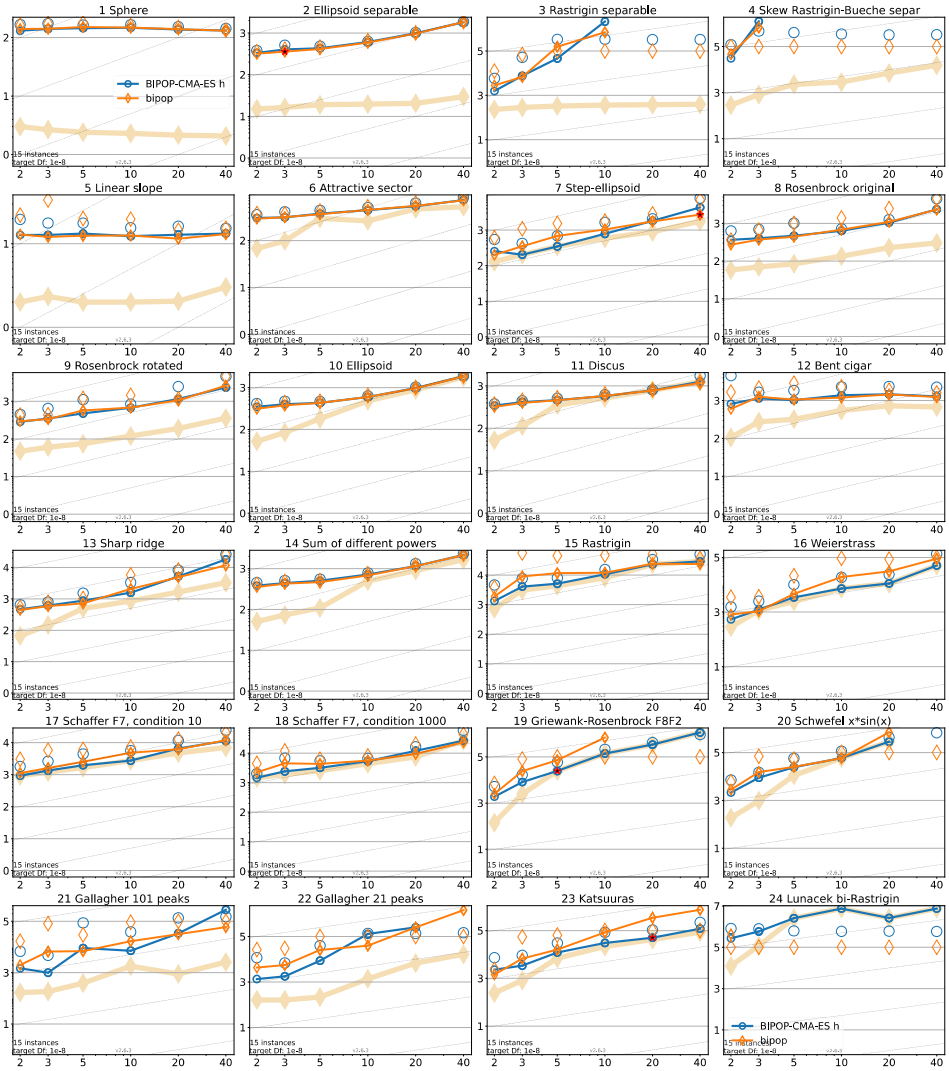
- Path: `opt\apop_cma_es.m`
- Standard cumulation constant values from `purecmaes_basic.m` (Hansen) are used with `cc = (4 + mueff/N) / (N+4 + 2*mueff/N)`, `cs = (mueff+2) / (N+mueff+5)` ([NH17] mentions slightly different values in Line (3) of their Algorithm)
- APOP is implemented as described in [OB24] with a simplified population increase/decrease scheme using a factor of 2, and a $\sqrt{\mu}$-law for σ-rescaling. [NH17] use more complicated ad-hoc expressions for population change and σ-rescaling
- [NH17] employ an additional restart scheme, i.e., they run CMA-ES once with constant N-dependent population size before the actual adaptive algorithm is started. This improves the performance of some, mostly unimodal, test functions, but masks the actual performance of the the APOP. Here, no additional restart scheme is used, which will explain some of the deviations observed compared to the CMAES-APOP of [NH17].
- [NH17] use additional and slightly modified stopping criteria. Here, we use the conditions from Sec. 2.2 of the simple CMA-ES.

The dynamics in Fig. 3 show expected performance from APOP (see also [OB24]), which is relatively well-balanced. On the sphere, it shows somewhat elevated λ-levels (sphere: $\lambda \approx 30$, ellipsoid: in parts $\lambda \approx 300$ and in the end $\lambda \approx 30$). It solves Rastrigin $N = 20$ and $N = 100$ by appropriately adapting λ.

The BBOB results in Fig. 4 show notable deviations between existing data (CMAES-APOP) and the new implementation, but the performances are not too far off. As mentioned above, [NH17] achieve better performance using a restart scheme before launching the actual APOP. As an example, the performance of CMAES-APOP [NH17] for f1, f5, f6, and f12 appears better due to the initial simple restart and is comparable to results of the simple CMA-ES (see Fig. 1). In general, [NH17] is mostly somewhat more efficient. Presumably, [NH17] performed more algorithm tuning for their results.

Overview page

Scaling of run "time" with dimension



Expected running time (ERT 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 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: ○: BIPOP-CMA-ES hansen noiseless, ◊: bipop

Figure 2: Comparing existing BIPOP data (blue) to new implementation (orange).

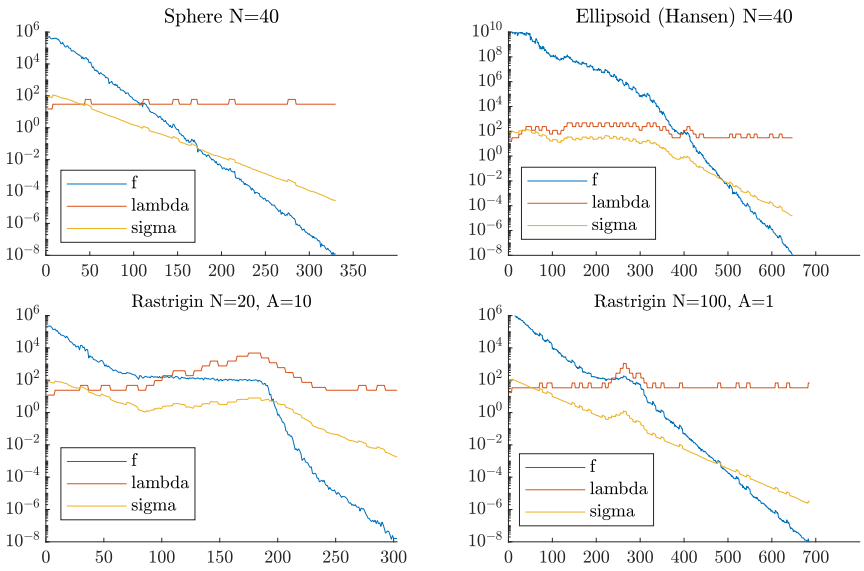
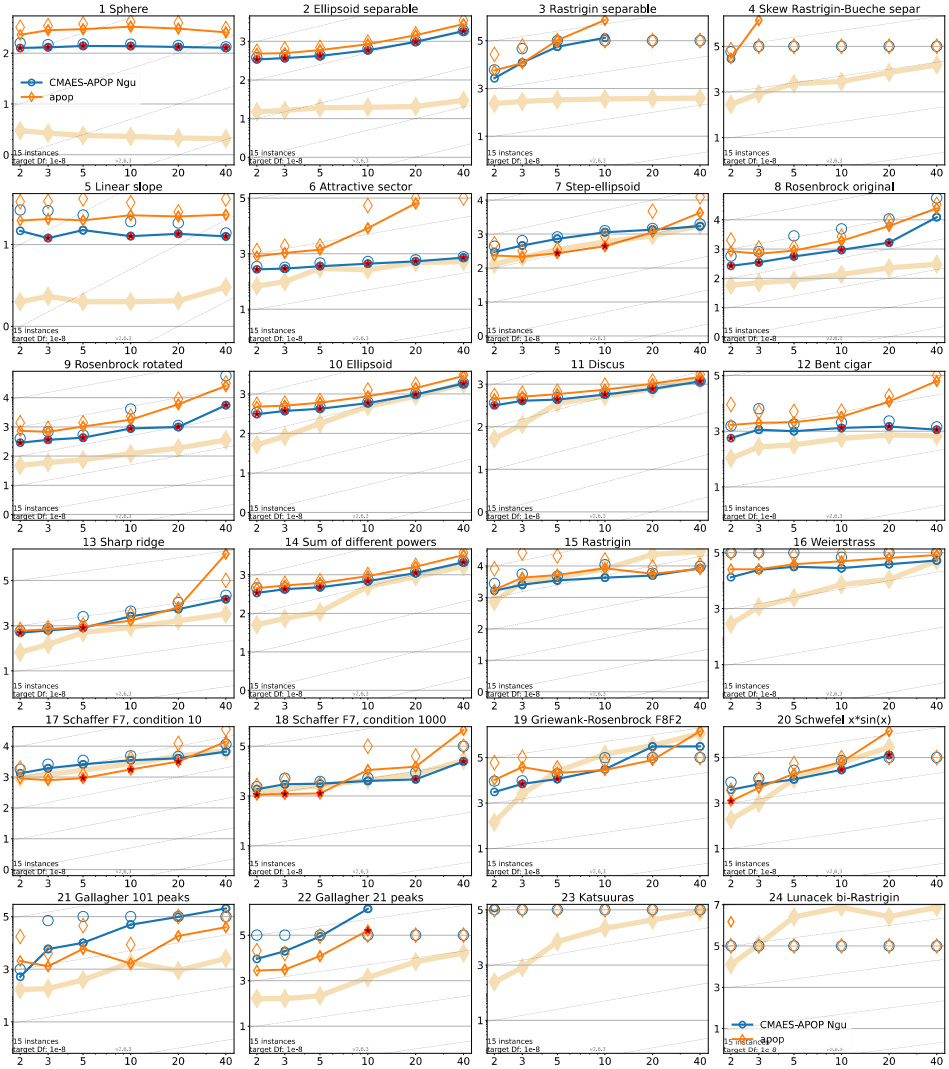


Figure 3: APOP dynamics on three functions (top: Sphere, Ellipsoid; bottom: Rastrigin). Initialization at $\mathbf{y}^{(0)} = 100 \cdot \mathbf{1}$, $\sigma^{(0)} = 100$, and $\lambda^{(0)} = \lambda_{\text{def}} = 4 + \lfloor 3 \log N \rfloor$.

Overview page

Scaling of run "time" with dimension



Expected running time (ERT 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 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: ○: CMAES-APOP Nguyen, ◊: apop

Figure 4: CMAES-APOP data (blue) compared to new implementation (orange).

2.5 PSA

PSA-CMA-ES from [NA18a] was implemented with the following simplifications

- Path: `opt\psa_cma_es.m`
- $h_\sigma = 1$ in rank-one update of covariance matrix permanently and not step-wise as shown in [NA18a, (5)]
- time-dependent normalization factors $\gamma_c = \gamma_\sigma = \gamma_\theta = 1$ for simplicity (see also comment above [NA18a, (8)], mentioning that their effect is barely recognizable)

- Similarly, the expected value of the Fisher-transformed update from [NA18a, (13)] is approximated as $\mathrm{E}\left[\left\|\mathcal{I}_\theta^{1/2}\Delta\theta\right\|^2\right] \simeq N/\mu_w$ based on the argumentation in [OB24]
- simplified σ-rescaling expression from [NA18a, (18)] by assuming $\mu_w N \gg \mu_w$. Hence, the coefficient of the weighted average of expected value of normal order statistics is not needed, such that the rescaling yields $\sigma^{(g+1)} = \sigma^{(g)}\mu_w^{(g+1)}/\mu_w^{(g)}$
- Here, we use the termination conditions from Sec. 2.2 of the simple CMA-ES.

The PSA dynamics in Fig. 5 show known elevated levels of $\lambda \approx 10^2$ on unimodal functions [OB24]. It shows good performance on Rastrigin for both $N = 20$ and $N = 100$, showing global convergence.

The BBOB results in Fig. 6 show close similarities of the existing data (PSA-CMA-ES) and the new implementation. Interestingly, the new implementation outperforms the original algorithm in multiple cases to some extent (e.g. f1, f16, f23) . However, in some cases the original algorithm is faster or more successful (f18, f19, f20). This can also be attributed to differences in the termination criteria. Given that the implementation and termination criteria were simplified, the results can be regarded as very good.

Note that an additional paper by the same authors [NA18b] focuses on more details on the implementation than the actual working principle of the PSA. In [NA18b] the authors also suggest performing an initial run of the basic CMA-ES to improve performance on simpler problems (similar to the approach in [NH17]).

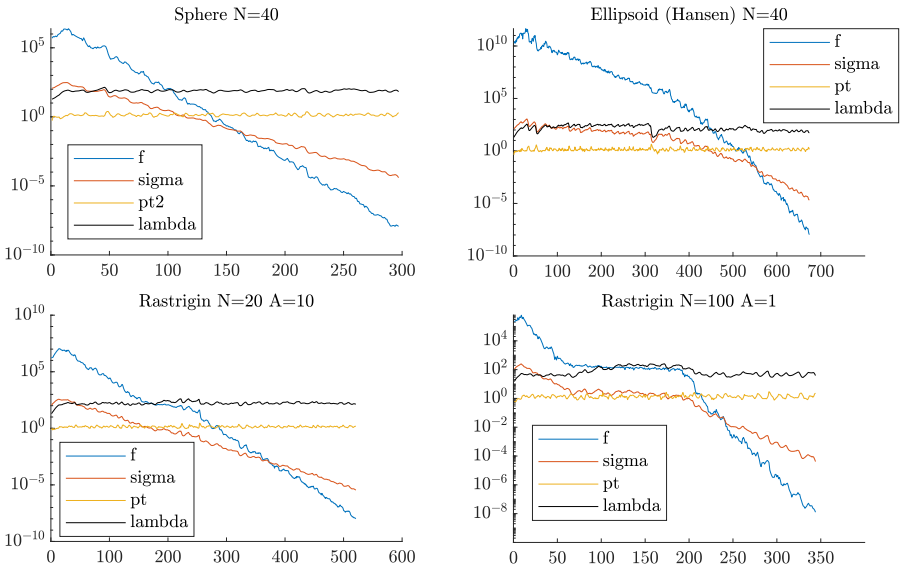


Figure 5: PSA-CMA-ES dynamics on three functions (top: Sphere, Ellipsoid; bottom: Rastrigin). Initialization at $\mathbf{y}^{(0)} = 100 \cdot \mathbf{1}$, $\sigma^{(0)} = 100$, and $\lambda^{(0)} = \lambda_{\mathrm{def}} = 4 + \lfloor 3 \log N \rfloor$.

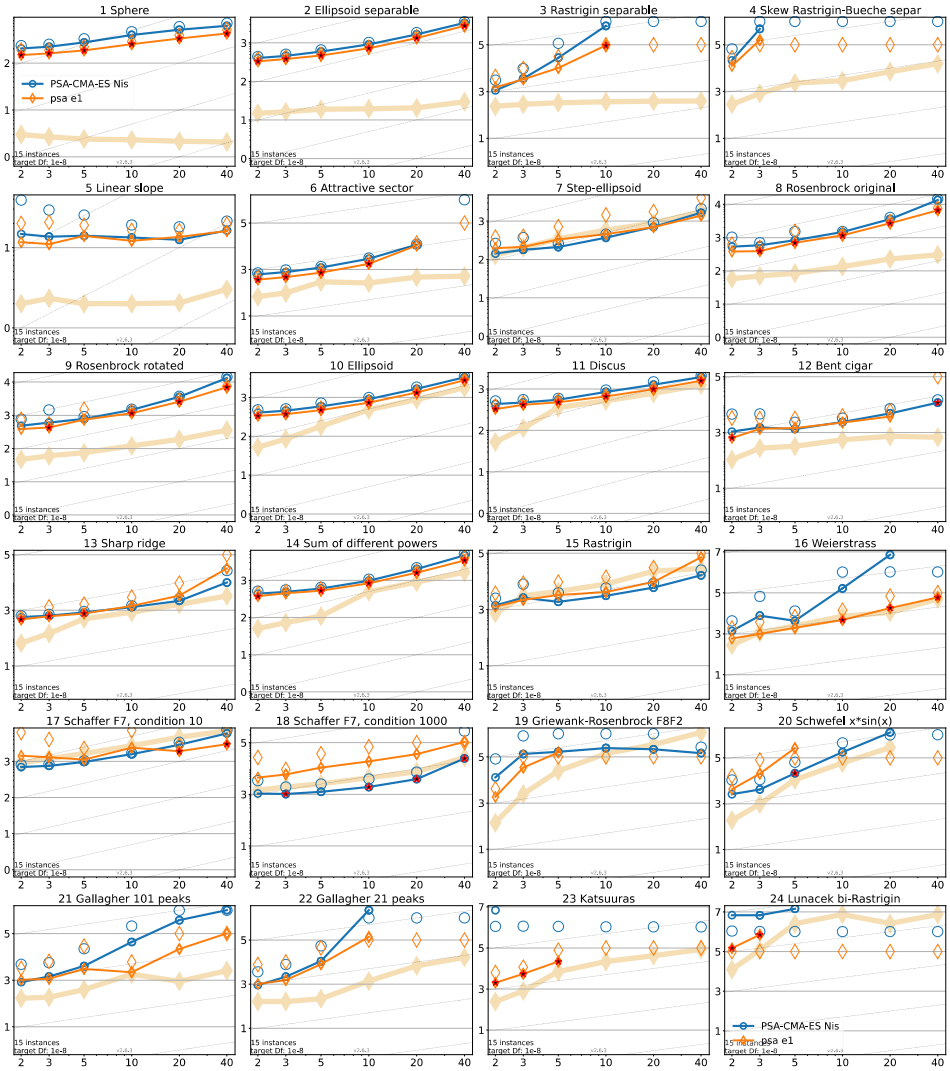
2.6 LRA and ETA

The LRA-CMA-ES was implemented according to [NAO23], which rescales the updates of the search vector ("mutate large and inherit small") and covariance matrix by the two factors η_m and η_Σ, respectively. The following remarks regarding the implementation of LRA can be given

- Path: `opt\lra_cma_es.m`
- The projection $\Pi_{[-1,1]}$ mentioned in [NAO23, (14)] was implemented as a cutoff of the argument $(SNR/(\alpha\eta - 1))$ above 1 and below -1, since they mention in the text "introduced to prevent a significant change of η in one iteration". Furthermore, they provide no additional information on the projection method.
- The vectorization operation of the Fisher-transformed update $\tilde{\Delta}_\Sigma$ was implemented by including all N^2 elements of the update (see `dS_tilde = reshape(dS_mat,N*N,1)` in code), since no additional

Overview page

Scaling of run "time" with dimension



Expected running time (ERT 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 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: ○: PSA-CMA-ES Nishida, ◊: psa e1

Figure 6: Existing PSA-CMA-ES data (blue) and new implementation (orange).

information was provided. This is in contrast to PSA, where only the diagonal and one set of off-diagonal elements are considered for vectorization (see `S_tilde = dS_mat(triu(dS_mat)~=0)`).

- It was observed that the value of σ from the rescaled Σ-update in [NAO23, Line 21] could reach zero during the determinant calculation, which could be an effect of numerical instabilities. This was observed on the sphere and ellipsoid for $N = 40$. The determinant also reached the value infinity on the Rastrigin function. Hence, the new implementation demands the determinant to be larger than zero and smaller than infinity before updating σ in [NAO23, Lines 21-23]. If the two conditions are not met, the σ-update is omitted.
- Here, we use the termination conditions from Sec. 2.2 of the simple CMA-ES.

Figure 7 shows the dynamics on three test functions. Comparing the new implementation with the results in [NAO23, Fig. 1] shows similar characteristics of η_m and η_Σ. η_m decreases on unimodal functions and reaches comparably small values (10^{-2} on the sphere, 10^{-3} on the ellipsoid). η_Σ oscillates mostly between 1 and 10^{-1}. The steady decrease of η_m is undesired since it significantly decreases the convergence speed on sphere and ellipsoid. The dynamics on the Rastrigin function ($N = 20$) are also similar to [NAO23, Fig. 1] ($N = 10$). η_m drops to roughly 10^{-3}, and then shows a minor increase, while η_Σ shows a prominent drop to roughly 10^{-2} during stagnation. On Rastrigin $N = 100$, η_m drops and remains very small even during global convergence, which decreases its efficiency significantly.

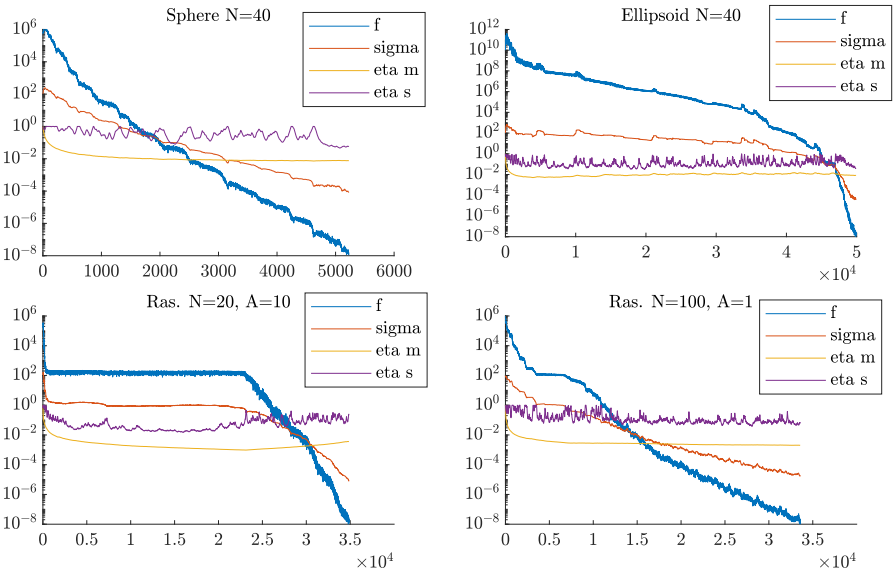


Figure 7: LRA-CMA-ES dynamics on three functions (top: Sphere, Ellipsoid; bottom: Rastrigin). Initialization at $\mathbf{y}^{(0)} = 100 \cdot \mathbf{1}$, $\sigma^{(0)} = 100$, $\lambda = \lambda_{\text{def}} = 4 + \lfloor 3 \log N \rfloor$, $\eta_m^{(0)} = 1 = \eta_\Sigma^{(0)}$.

2.6.1 ETA (simplified LRA)

The results discussed in Fig. 7 suggest testing a simplified version of the LRA, called ETA, under the following assumptions:

- Path: `opt\eta_cma_es.m`
- the learning rate η should decrease when insufficient algorithm performance is detected
- Fig. 7 shows that η decreases on unimodal functions, which is not necessary.
- Hence, the simpler performance measure of APOP from Sec. 2.4 is implemented to tune only the learning rate η_m of the mean vector update. The rescaled update of the covariance matrix and σ of [NAO23], which showed numerical instabilities in some cases, is omitted.
- The presented implementation is closely related to APOP in [OB24] as follows.

- ETA evaluates the differences of the median of selected fitness values over $L = 10$ generations with `df = diff(f_med_g)`, and measures the ES performance by counting the number of fitness deteriorations `P = sum(df>0)/(L-1)`. Using the threshold `P_Thr = 4/(L-1) = 4/9` and an increase factor `FACTOR_ETA = 1.5`, we update

```
eta_m(g) = eta_m(g-1):  P = P_Thr
eta_m(g) = eta_m(g-1)*FACTOR_ETA: P < P_Thr
eta_m(g) = eta_m(g-1)/FACTOR_ETA: P > P_Thr
```

- The relatively high `P_Thr=4/9` is useful experimentally, since one observes a significant number of fitness deteriorations with `eta_m < 1` even on unimodal functions. However, tuning the threshold is a crucial point whether population size or learning rate are adapted.
- Additionally, `0.01 < eta_m < 1` is ensured, where the lower bound ensures that the ES does not become "too slow"
- if `eta_m` is updated, linear σ rescaling `sigma = sigma*(eta_m(g-1)/eta_m(g))` is performed as suggested by [NAO23].
- Again, the termination conditions from Sec. 2.2 of the simple CMA-ES are used

Figure 8 shows the dynamics on sphere, ellipsoid, and Rastrigin. On the right, the performance measure of the ETA is displayed.

- η_m remains small on unimodal functions and increases on the Rastrigin function during the slow-down.
- After the slow-down, η_m increases again to maintain efficiency. In the given example, ETA appears unsuccessful on Rastrigin. For BBOB in Fig. 9, however, it will show better results on Rastrigin for $N \leq 40$. A more detailed study of ETA for various dimensionalities evaluating success rate and expected runtime could be useful.

Figure 9 shows BBOB results of LRA-, ETA-, APOP-, and PSA-CMA-ES for comparison.

- ETA shows improved performance compared to LRA, especially on unimodal functions. Exceptions are f15, f17, and f18. LRA becomes notably more inefficient for larger N, which is likely related to the way it controls η. In Fig. 8 it was observed that η decreased and remained significantly low.
- In multiple cases, ETA shows a significant performance increase (f1, f3, f16, f23).
- Comparing ETA with APOP, and ETA with PSA, one observes somewhat comparable performance, where one method is faster than the other one in some cases, and vice-versa.
- These observations suggest a more detailed comparison. In particular, it may be useful to study the dynamics of λ and η adaptation in more detail under exactly the same performance measure.
- The question of choosing the default (N-dependent?) population size for η-adaptation also remains an open task. In Fig. 8, $\lambda = N$ is chosen as a starting point.

Note that [NAO23] did not include BBOB results, such that a direct comparison is not possible. Furthermore, [NAO23] use the success performance measure SP1 from [AH05] in their analysis, which includes only the function evaluations of successful runs and excludes unsuccessful runs. Standard BBOB evaluates SP2, see [HABT22], including function evaluations of unsuccessful runs.

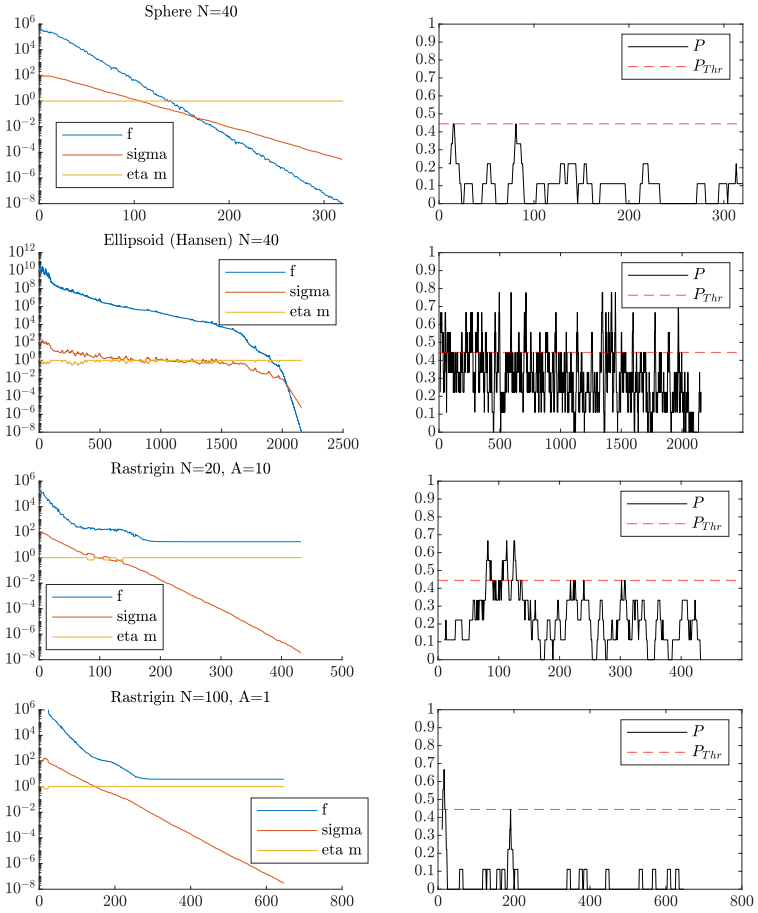
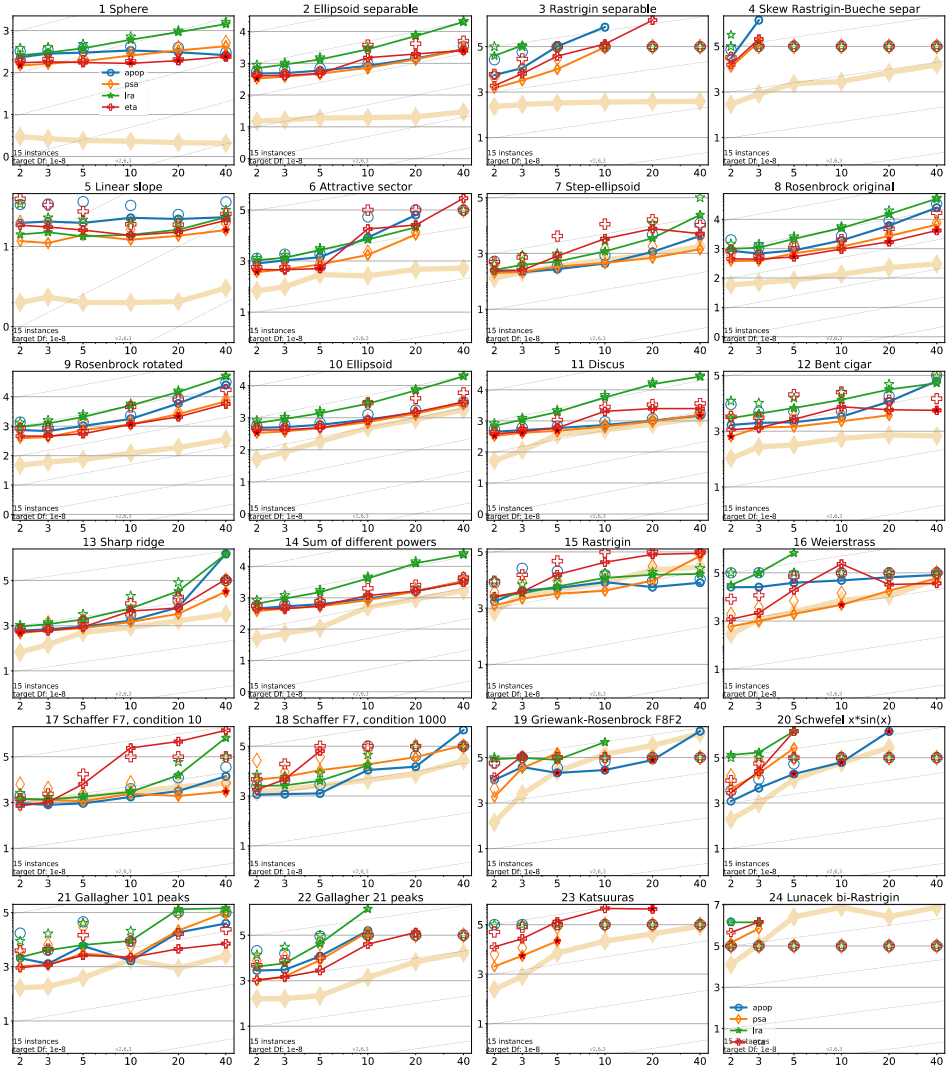


Figure 8: ETA-CMA-ES (simplified LRA) dynamics on three functions (sphere, ellipsoid, Rastrigin. Initialization at $\mathbf{y}^{(0)} = 100 \cdot \mathbf{1}$, $\sigma^{(0)} = 100$, and $\lambda = N$.

Overview page

Scaling of run "time" with dimension



Expected running time (ERT 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 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: ○: apop, ◊: psa, ☆: lra, +: eta

Figure 9: LRA and ETA compared to APOP and PSA implementations.

2.7 Meta-CMSA-ES

Motivated by the BIPOP-CMA-ES implementation, an alternative approach is to implement a Meta-ES with covariance matrix adaptation that tests two population sizes consecutively and selects the more successful run for the next generation parameter values. A self-adaptive ES is the preferred choice since no cumulation paths (obtained with different population sizes) need to be passed between Meta-ES-generations. To this end, a CMSA-ES from [BS08] is implemented as the inner ES (see `opt/subroutines/cmsa_es_meta.m` in code), while the outer ES is implemented in `opt/meta_es.m`. The following ideas are applied:

- outer ES isolates two CMSA-ES with increased (`lambda2`) and decreased (`lambda1`) population size λ, respectively, by a factor `POP_FACTOR = 2`. One sets, e.g., `LAMBDA_MIN = 8` and `LAMBDA_MIN = 4096`.

- Isolation must be done w.r.t. a budget of function evaluations. With isolation over a fixed number of generations, λ is unnecessarily increased on unimodal functions.
- Isolation was implemented as `STOP_INNER.FEVAL_MAX = ISO_FACTOR*lambda2`, where `ISO_FACTOR=10` and `lambda2` is the larger population size. This ensures that the larger population runs over multiple generations (`ISO_FACTOR` generations) before comparing it with the smaller population size. Hence, the smaller population size runs for 40 generations (`2*POP_FACTOR*ISO_FACTOR = 40`).

The inner ES works as follows:

- Inputs: increased/decreased population size μ and λ ($\vartheta = 1/2$), rescaled σ (decrease: `sigma/r_sigma`, increase: `sigma*r_sigma` with `r_sigma = sqrt(POP_FACTOR)` using $\sqrt{\mu}$-rescaling law from [OB24]).
- previously learned covariance matrix `C` is given as input without modifications.
- Outputs: fitness `f`, vector of the parental centroid `y`, mutation strength `sigma`, covariance matrix `C`, and the square-root of minimum (`min_D`) and maximum (`max_D`) eigenvalues for checking the termination criteria. To reduce local attraction effects in some cases (e.g. Rastrigin), it is important to output the parental centroid and not the overall best individual. However, this introduces one additional function evaluation.

Figure 10 shows the dynamics of the Meta-CMSA-ES.

- On the sphere and ellipsoid, it reduces its initial population size (slightly elevated) to smaller value. Mostly, $\lambda = 8, 16$ is observed. The results on the sphere are promising in terms of efficiency (see generation number). However, the number of generations on the ellipsoid is very high, which indicates issues in learning the covariance matrix through self-adaptation and Meta-ES.
- On Rastrigin, one observes an increase for $N = 20$, which is good. However, local convergence is observed.
- For Rastrigin $N = 100$, Meta-ES rapidly decreases the population size and local convergence occurs.

It appears that the local search of the smaller population wins over the more global search of the larger population by finding a local optimum, and passing this result over to the next isolation period. The suboptimal results hold also with multiple repetitions, showing a success rate $P_S \gtrapprox 0$. The results on Rastrigin can be improved by isolating for a number of generations, e.g. 10, instead of function evaluations. However, this increases the overall population size level, not only on Rastrigin, but also on the sphere.

Figure 11 shows BBOB benchmarking results:

- Meta-CMSA-ES is compared to existing BIPOP-CMA-ES data (Hansen)
- Nearly all ERT of the Meta-ES are worse compared to BIPOP-CMA-ES.
- ERT of Meta-ES tends to increase significantly for increasing N, showing major inefficiency. This was also observed in Fig. 10.
- Presumably, the simultaneous evaluation of multiple population sizes is less efficient than repeated, well-controlled restarts both for unimodal problems and for multimodal problems. Furthermore, restarts are useful under multimodality to discard solutions in local attractors, which can survive in a Meta-ES run.
- The results could be related to properties of the self-adaptation. However, a more detailed comparison between a simple CMSA-ES and Meta-CMSA-ES could also yield more information.

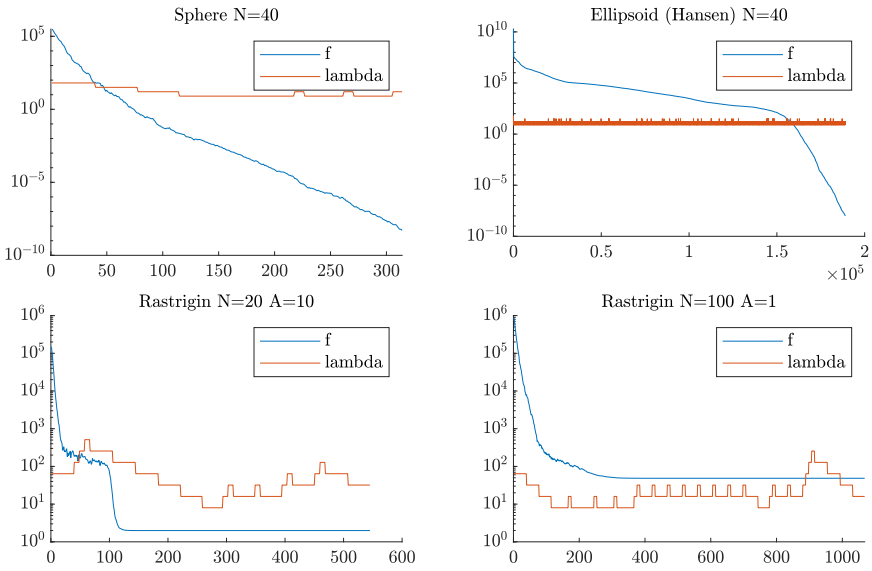
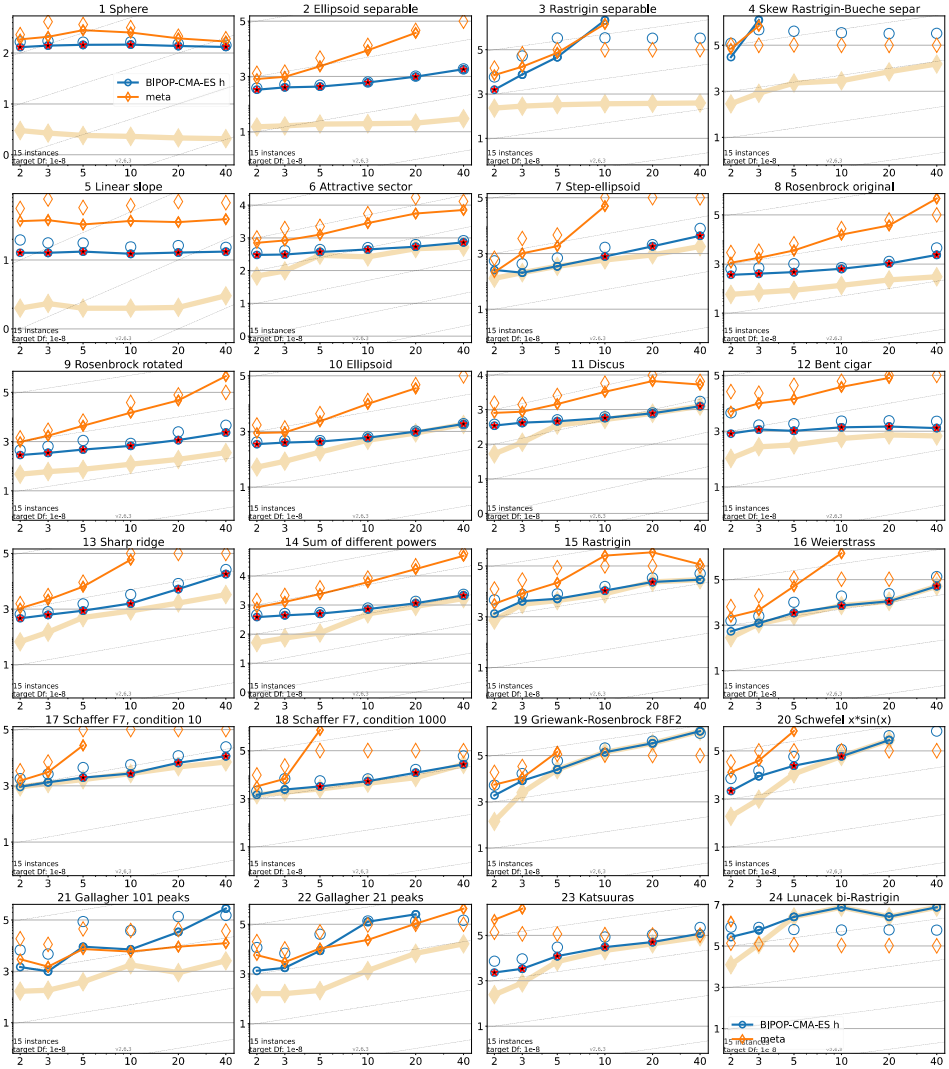


Figure 10: Meta-CMSA-ES dynamics on three functions (top: Sphere, Ellipsoid; bottom: Rastrigin). Initialization at $\mathbf{y}^{(0)} = 100 \cdot \mathbf{1}$, $\sigma^{(0)} = 100$, and $\lambda^{(0)} = 128$. The f- and λ-dynamics are concatenated from the respective selected inner runs.

Overview page

Scaling of run "time" with dimension



Expected running time (ERT 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 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: ○: BIPOP-CMA-ES hansen noiseless, ◊: meta

Figure 11: Meta-CMSA-ES compared to BIPOP-CMA-ES by Hansen.

3 Preliminary Conclusions

From the experiments of the previous sections, one could arrive at the following conclusions:

- The most promising performance was achieved by APOP, BIPOP, ETA, and PSA. Hence, the overall aggregated performance will be displayed in Fig. 12.
- The Meta-CMSA-ES in the presented configuration does not appear as an efficient algorithm choice compared to, e.g., the BIPOP-CMA-ES.
- The LRA-CMA-ES from [NAO23] does yield viable BBOB results, but its efficiency decreases significantly with increasing N. This appears to be related to its performance measure (a signal-

to-noise ratio), which reduces η_m and η_Σ notably, even on unimodal functions, see Fig. 7. However, it may be useful to study its implementation in greater detail.

To gain an overall picture of the performance by aggregating all functions and targets for given dimension, one can study the BBOB runtime distribution in Fig. 12:

- For small dimensionalities $N = 2, 3, 5$, there is no clear winner. Hence, one could use the simple CMA-ES implementation of Sec. 2.2 ("pure") to obtain competitive performance. Simple restarting of the ES with relatively small default population size appears to be a valid strategy for small N and no advanced adaptation methods are necessary.

- As larger $N = 10, 20, 40$ are tested, more differences between the methods can be observed. The basic ("pure") CMA-ES deteriorates notably, indicating that advanced techniques should be used. In general, BIPOP shows the highest number of reached targets. However, BIPOP is probably heavily tuned to the BBOB testbed. Whether its relatively good performance holds for higher N and other test functions is an open question.

- APOP, ETA, and PSA appear relatively close, with PSA showing slightly worse results for $N = 40$. Interestingly, APOP and ETA are close in terms of performance. They both use the same type of performance measure, but with slightly different parameter settings. With insufficient ES performance, APOP increases λ and ETA decreases its learning rate η_m, improving the global search performance.

- These first results indicate that both population size and learning rate adaptation can reach comparable overall benchmark results, provided that the underlying performance measures are comparable.

- Furthermore, the suggested ETA-CMA-ES (simplified LRA from [NAO23]) appears as a viable alternative to the LRA-CMA-ES given the first benchmark tests in this report. It could be interesting to test a learning rate η-restart scheme (similar to BIPOP) and compare it to an adaptive η control algorithm.

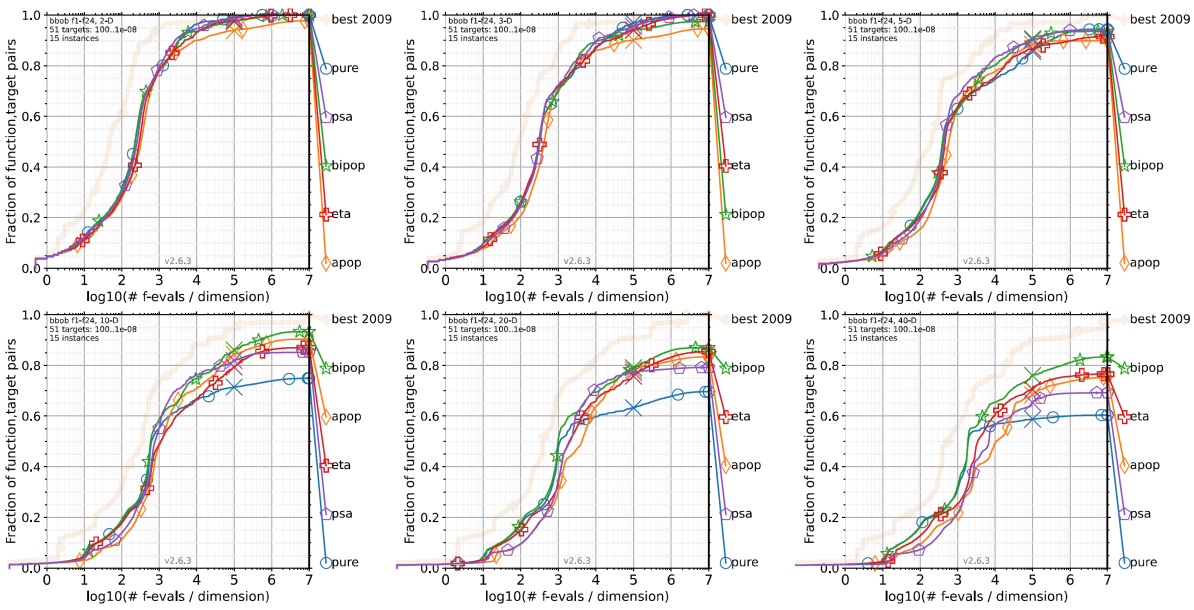


Figure 12: BBOB runtime distributions (ECDFs) over all targets.

References

[AH05] A. Auger and N. Hansen. A Restart CMA Evolution Strategy with Increasing Population Size. In *Congress on Evolutionary Computation, CEC'05*, volume 2, pages 1769–1776. IEEE, 2005.

[Bey98] H.-G. Beyer. Mutate Large, But Inherit Small! On the Analysis of Rescaled Mutations in $(\tilde{1}, \tilde{\lambda})$-ES with Noisy Fitness Data. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, 5*, pages 109–118, Heidelberg, 1998. Springer.

[BS08] H.-G. Beyer and B. Sendhoff. Covariance Matrix Adaptation Revisited – the CMSA Evolution Strategy. In G. Rudolph et al., editor, *Parallel Problem Solving from Nature 10*, pages 123–132, Berlin, 2008. Springer. DOI: 10.1007/978-3-540-87700-4_13.

[BS17] H.-G. Beyer and B. Sendhoff. Simplify Your Covariance Matrix Adaptation Evolution Strategy. *IEEE Transactions on Evolutionary Computation*, 21(5):746–759, 2017. DOI: 10.1109/TEVC.2017.2680320.

[HABT22] N. Hansen, A. Auger, D. Brockhoff, and T. Tušar. Anytime performance assessment in blackbox optimization benchmarking. *IEEE Transactions on Evolutionary Computation*, 26(6):1293–1305, 2022.

[Han09] N. Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Noisy Testbed. In *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 2397–2402. ACM, July 2009.

[NA16] K. Nishida and Y. Akimoto. Population Size Adaptation for the CMA-ES Based on the Estimation Accuracy of the Natural Gradient. In *GECCO'16: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 237–244, New York, 2016. ACM.

[NA18a] K. Nishida and Y. Akimoto. PSA-CMA-ES: CMA-ES with Population Size Adaptation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 865–872, New York, NY, USA, 2018. Association for Computing Machinery.

[NA18b] Kouhei Nishida and Youhei Akimoto. Benchmarking the psa-cma-es on the bbob noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, page 1529–1536, New York, NY, USA, 2018. Association for Computing Machinery.

[NAO23] M. Nomura, Y. Akimoto, and I. Ono. CMA-ES with Learning Rate Adaptation: Can CMA-ES with Default Population Size Solve Multimodal and Noisy Problems? In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 839–847, New York, NY, USA, 2023.

[NH17] D. M. Nguyen and N. Hansen. Benchmarking CMAES-APOP on the BBOB Noiseless Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, page 1756–1763, New York, NY, USA, 2017. Association for Computing Machinery.

[OB24] A. Omeradzic and H.-G. Beyer. On the Interaction of Adaptive Population Control with Cumulative Step-Size Adaptation, 2024. Available at: https://arxiv.org/abs/2410.00595.