LB+IC-CMA-ES: Two Simple Modifications of CMA-ES to Handle Mixed-Integer Problems

Tristan Marty[1,2], Nikolaus Hansen[2(✉)], Anne Auger[2], Yann Semet[1], and Sébastien Héron[1]

[1] Thales Research and Technology, Palaiseau, France
[2] Inria and CMAP, Ecole Polytechnique, IP Paris, Palaiseau, France
nikolaus.hansen@inria.fr

Abstract. We present LB+IC-CMA-ES, a variant of CMA-ES that handles mixed-integer problems. The algorithm uses two simple mechanisms to handle integer variables: (i) a lower bound (LB) on the variance of integer variables and (ii) integer centering (IC) of variables to their domain middle depending on their value. After presenting the algorithm, we evaluate the different variants ensuing from these modifications on the BBOB mixed-integer testbed and compare the performance with the recently introduced CMA-ES with margin.

Keywords: mixed-integer optimization · CMA-ES · Evolution Strategies

1 Introduction

Mixed-integer optimization problems appear commonly in applications. Applying a continuous optimizer, for example the CMA-ES algorithm [5], *as is* to mixed-integer problems often leads to premature convergence of the integer coordinates on the wrong plateau. For this reason, different variants of CMA-ES have been proposed in recent years to handle mixed-integer problems, including CMA-ES with margin for the $(\mu/\mu_w, \lambda)$-CMA-ES [3] and also for the $(1+1)$-CMA-ES [14]. In order to prevent stagnation, the CMA-ES with margin sets lower bounds on the marginal probabilities to mutate integer variables. Other Evolution Strategies to handle mixed-integer problems include DX-NES-ICI [10] which has been designed specifically for mixed-integer problems where continuous variables are more decisive than integer ones, or MIESs [11] that uses a different mutation operator for non-continuous variables.

In this context, we build on a version of CMA-ES that bounds the variance of integer coordinates from below [12]. First, we propose a theoretically motivated setting of the lower bound. Second, we introduce the centering of mutated integer variable values to the middle of the integer plateau. This latter mechanism usually introduces a bias on the sample population average for which we consequently correct.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 284–299, 2024.
https://doi.org/10.1007/978-3-031-70068-2_18

The paper is organized as follows. Section 2 presents CMA-ES and the two modifications introduced to improve the performance on mixed-integer problems. Section 3 illustrates the impact of both mechanisms by showing single runs on an ill-conditioned ellipsoid function with some integer variables. Section 4 assesses the performance of the different variants on the mixed-integer BBOB testbed as well as on functions with varying fraction of integer variables and Sect. 5 concludes the paper.

2 Two Simple Modifications of CMA-ES to Handle Mixed-Integer Problems

We describe the CMA-ES algorithm for numerical minimization of a function $f : \mathbb{R}^N \to \mathbb{R}$ and our two modifications to handle mixed-integer problems.

When given a mixed-integer function $f(\boldsymbol{x})$ where some coordinates of $\boldsymbol{x}$ belong to $\mathbb{Z}$ and other coordinates belong to $\mathbb{R}$, we denote by $\mathcal{S}_\mathbb{Z}$ the index set of integer coordinates and $\mathcal{S}_\mathbb{R}$ the index set of real coordinates. The cardinality of $\mathcal{S}_\mathbb{Z}$ equals $N_{\text{int}} = |\mathcal{S}_\mathbb{Z}|$ and we have $N = |\mathcal{S}_\mathbb{Z}| + |\mathcal{S}_\mathbb{R}|$. Then, $x_i \in \mathbb{Z}$ for $i \in \mathcal{S}_\mathbb{Z}$ and $x_i \in \mathbb{R}$ for $i \in \mathcal{S}_\mathbb{R}$. We define the function

$$\text{int}[.] : \mathbb{R} \to \mathbb{Z}, x \mapsto \lfloor x + 1/2 \rfloor \tag{1}$$

yielding the "integer value" of $x \in \mathbb{R}$. When we apply a continuous algorithm to a mixed-integer function, we apply int[.] to each integer coordinate before we evaluate f.

2.1 CMA-ES

The $(\mu/\mu_w, \lambda)$-CMA-ES samples λ solutions $\boldsymbol{x}$ at each iteration t distributed according to a multivariate normal distribution $\boldsymbol{x}_i^{(t)} \sim \mathcal{N}(\boldsymbol{m}^{(t)}, (\sigma^{(t)})^2 \mathbf{D}^{(t)} \mathbf{C}^{(t)} \mathbf{D}^{(t)})$ where $\boldsymbol{m}^{(t)} \in \mathbb{R}^N$ represents the incumbent mean solution. The covariance matrix of the sampling distribution is decomposed into three parts: (i) the overall step-size $\sigma^{(t)}$; (ii) the covariance matrix $\mathbf{C}^{(t)}$ containing information about the sensitivity in some principal axes; (iii) the diagonal matrix $\mathbf{D}^{(t)}$ introduced in [1] to scale the distribution in the given coordinate system.

For sampling λ solutions, we transform samples from an isotropic normal distribution, $\boldsymbol{z}_i^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, using $\mathbf{D}^{(t)}$ and $\mathbf{C}^{(t)}$,

$$\boldsymbol{y}_i^{(t)} = \sqrt{\mathbf{C}^{(t)}} \boldsymbol{z}_i^{(t)} \ ; \ \boldsymbol{x}_i^{(t)} = \boldsymbol{m}^{(t)} + \sigma^{(t)} \mathbf{D}^{(t)} \boldsymbol{y}_i^{(t)} \quad \text{for } i = 1, \ldots, \lambda \ , \tag{2}$$

where $\sqrt{\mathbf{C}^{(t)}}$ is symmetric and positive definite. The candidate solutions are ranked according to their objective function f. The index $i{:}\lambda$, as defined by the next equation, refers to the i^{th} solution when ordered by their f-value,

$$f(\boldsymbol{x}_{1:\lambda}^{(t)}) \leq f(\boldsymbol{x}_{2:\lambda}^{(t)}) \leq \ldots \leq f(\boldsymbol{x}_{\lambda:\lambda}^{(t)}) \ . \tag{3}$$

The new mean is a weighted recombination of the sorted solutions

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \sigma^{(t)}\mathbf{D}^{(t)} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \tag{4}$$

with $\mu = \lfloor \lambda/2 \rfloor$ and $w_i \propto \log((\lambda+1)/2) - \log(i)$ such that $\sum_{i=1}^{\mu} w_i = 1$.

Step-Size. The parameter $\sigma^{(t)}$ is updated using the length of a so-called evolution path, $\boldsymbol{p}_\sigma^{(t+1)} \in \mathbb{R}^N$. The step size is increased if $\|\boldsymbol{p}_\sigma^{(t+1)}\|$ is larger than the expected length of a standard normally distributed random vector and decreased if it is smaller, specifically

$$\boldsymbol{p}_\sigma^{(t+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(t)^{-\frac{1}{2}}} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \tag{5}$$

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\mathbb{E}\left(\|\mathcal{N}(\mathbf{0},\mathbf{I})\|\right)}\right)\right) \quad , \tag{6}$$

where c_σ determines the decay of $\boldsymbol{p}$ and, by default, $c_\sigma = \frac{\mu_{\text{eff}}+2}{N+\mu_{\text{eff}}+5}$ with $\mu_{\text{eff}} = 1/\sum_i w_i^2$ and the step-size damping $d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}}-1}{N+1}} - 1\right) + c_\sigma$.

Covariance Matrix. Finally, the covariance matrix $\mathbf{C}^{(t)}$ is updated with both rank-one and rank-μ updates, the latter of which makes use of negative weights. Like for the step size update, the covariance matrix update relies on an evolution path updated as

$$\boldsymbol{p}_c^{(t+1)} = (1-c_c)\boldsymbol{p}_c^{(t)} + h_\sigma \sqrt{c_c(2-c_c)\mu_{\text{eff}}} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \quad , \tag{7}$$

where h_σ prevents a rapid increase of variances in $\mathbf{C}^{(t)}$ when the step-size is already increasing, for example in the first iterations, and reads

$$h_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|^2}{1-(1-c_\sigma)^{2t}} < 2N\left(1 + \frac{2}{N+1}\right) \\ 0 & \text{otherwise} \end{cases} \quad . \tag{8}$$

Given the weights $w_1 \geq w_2 \geq \ldots \geq w_\mu > 0 \geq w_{\mu+1} \geq w_\lambda$ [6, Appendix A], we introduce

$$w_i^\circ = \begin{cases} w_i & \text{if } w_i \geq 0 \\ w_i \frac{N}{\|\sqrt{\mathbf{C}^{(t)^{-1}}}\boldsymbol{y}_{i:\lambda}\|^2} & \text{otherwise} \end{cases} \quad . \tag{9}$$

To guaranty positive definiteness, we further scale down the negative weights when necessary [1, Sec 3.2]. The covariance matrix update then reads:

$$\mathbf{C}^{(t+1)} = \left(1 + c_1(1-h_\sigma)c_c(2-c_c) - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i\right)\mathbf{C}^{(t)} + c_1 \boldsymbol{p}_c^{(t+1)}(\boldsymbol{p}_c^{(t+1)})^\top + c_\mu \sum_{i=1}^{\lambda} w_i^\circ \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\top \quad . \tag{10}$$

Learning rates and the cumulation parameter are set to $c_1 = \frac{2}{(N+1.3)^2+\mu_{\text{eff}}}$, $c_\mu = \min\left(1-c_1,\ 2\frac{\mu_{\text{eff}}+1/\mu_{\text{eff}}-1.75}{(N+2)^2+\mu_{\text{eff}}}\right)$ and $c_c = \frac{4+\mu_{\text{eff}}/N}{N+4+2\mu_{\text{eff}}/N}$.

The diagonal matrix $\mathbf{D}^{(t)}$ is not updated here but will be utilized to set a lower bound for integer variables.

2.2 Lower Bounding the Standard Deviation on Integer Coordinates

Applying CMA-ES as presented in the previous section to mixed-integer functions can lead to premature convergence of integer variables at a nonoptimal value when their standard deviation gets much smaller than 1/2. As the step-size converges to zero, also the probability to sample a solution with a different integer value converges to zero and the variable is trapped. This behavior can be observed in Figure 3a which shows the coordinates of the mean $\boldsymbol{m}^{(t)}$ during the optimization of a 30-dimensional mixed-integer ellipsoid function. We observe that four integer variables converge to a nonoptimal value.

To prevent the premature convergence of integer coordinates, we impose a lower bound, σ_{LB}, on their standard deviation by updating the diagonal matrix $\mathbf{D}^{(t)}$. Let $\cdot_{j,j}$ denote the j^{th} diagonal element of a matrix. We define

$$\sigma_{\text{std(j)}}^{(t)} = \sigma^{(t)}\mathbf{D}_{j,j}^{(t)}\sqrt{\mathbf{C}_{j,j}^{(t)}} \quad \text{for } j = 1,\ldots,N \ , \tag{11}$$

and update the diagonal matrix right before Eq. (2) as

$$\mathbf{D}_{j,j}^{(t)} \leftarrow \frac{\max\left(\sigma_{\text{LB}}, \sigma_{\text{std(j)}}^{(t)}\right)}{\sigma^{(t)}\sqrt{\mathbf{C}_{j,j}^{(t)}}} \quad \text{for } j \in \mathcal{S}_{\mathbb{Z}} \tag{12}$$

which changes $\mathbf{D}_{j,j}^{(t)}$ only when $\sigma_{\text{std(j)}}^{(t)} < \sigma_{\text{LB}}$ and ensures that $\sigma_{\text{std(j)}}^{(t)} \geq \sigma_{\text{LB}}$.

Setting the Lower-Bound. The choice of σ_{LB} indirectly controls the minimal mutation rate of integer variables. In order to allow the mutation rate to be small enough to not disrupt the optimization of continuous variables, we propose the lower bound

$$\sigma_{\text{LB}} = \min\left(\frac{\mu_{\text{eff}}}{N}, 0.2\right). \tag{13}$$

We estimate the effect of σ_{LB} on the *mutation rate* (the proportion of candidate solutions which is on a different integer plateau than the mean) based on the following simulation. We optimize the one-dimensional function $x \mapsto \lfloor x + 1/2 \rfloor^2$ with a fixed step-size starting at $x = 0$ and record at each iteration the fraction of solutions which are outside of $[-0.5, 0.5]$. The average is taken between iteration $3t_0$ and $\max(100\,t_0, 10^4/2)$ or at most 10^7, where t_0 is the first iteration with one mutation, i.e., with one candidate solution sampled outside $[-0.5, 0.5]$. The mutation rate, p_1, is shown in Fig. 1 for different step-sizes and different population sizes.

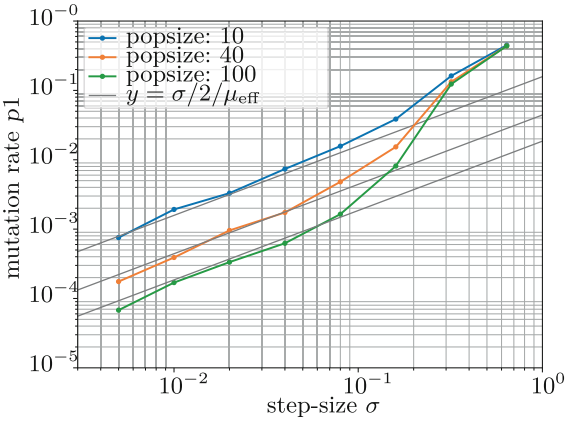


Fig. 1. Fraction of mutated individuals per iteration versus the step-size of the sample distribution for different population sizes.

The mutation rate can be well approximated by $p_1 \approx \sigma/\mu_{\mathrm{eff}}/2$ when $\sigma < 0.1$. This dependency on μ_{eff} is somewhat surprising, because the speed of the random walk is proportional to $1/\sqrt{\mu_{\mathrm{eff}}}$.

We now derive (13). Let p_N denote the probability that at least one integer coordinate of a solution is mutated (integer-different from the mean) and λ_{EC} denote the *effective continuous population size*, that is, the average number of solutions that are unaffected from integer mutations (that have in all integer coordinates the same integer value as the mean). Given that $\Delta f \to 0$ in the continuous subspace when $\sigma \to 0$, f-changes induced by an integer mutation dominate the f-values when σ is small. Therefore, to ensure progress in the continuous subspace, we want several solutions to be unaffected from integer mutations.

We have the following equations

$$p_1 \approx \frac{\sigma}{2\mu_{\text{eff}}} \qquad \text{if } \sigma < 0.1 \tag{14}$$

$$p_N \approx p_1 \times N_{\text{int}} \qquad \text{if } p_N \leq 1/2 \text{ or } \sigma \lesssim \mu_{\text{eff}}/N_{\text{int}} \tag{15}$$

$$\lambda_{\text{EC}} \approx \lambda(1 - p_N) \tag{16}$$

$$\lambda_{\text{EC}} \gtrsim \gamma \frac{N_{\text{int}}}{N} + \frac{N - N_{\text{int}}}{N}\lambda \qquad \text{if } N_{\text{int}} < N \tag{17}$$

Equation (14) follows from Fig. 1. Equation (15) approximates, when p_1 is small, $1 - p_N = (1 - p_1)^{N_{\text{int}}}$ assuming independence. Equation (16) approximates the unaffected population size, λ_{EC}. Equation (17) gives a heuristic lower bound for λ_{EC}, where γ represents the smallest reasonable population size for the continuous subspace (when N_{int} is close to N).

Combining the first two and the last two equations yields, respectively,

$$p_N \approx \frac{\sigma N_{\text{int}}}{2\mu_{\text{eff}}} \quad \text{and} \quad p_N \lesssim \frac{N_{\text{int}}}{N}\left(1 - \frac{\gamma}{\lambda}\right) \ , \tag{18}$$

and combining these gives an upper bound for the step-size

$$\sigma \lesssim \frac{2\mu_{\text{eff}}}{N}\left(1 - \frac{\gamma}{\lambda}\right) \quad \text{and} \quad \sigma \lesssim \frac{4\mu_{\text{eff}}}{3N} \quad \text{for } \gamma \lesssim \lambda/3 \ . \tag{19}$$

Equation (19) reveals the largest step-size that presumably allows for an effective search in the continuous subspace and is hence an upper bound for σ_{LB}.

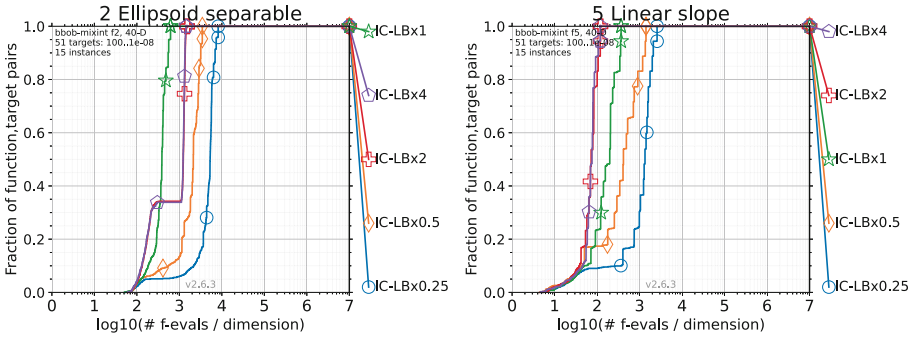


Fig. 2. Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 40. (Legend: algorithm IC-LBx0.25, IC-LBx0.5, IC-LBx1, IC-LBx2, IC-LBx4 are CMA-ES with integer centering and a lower bound defined $\sigma_{\text{LB}} = \min\left(\alpha\frac{\mu_{\text{eff}}}{N}, 0.2\right)$ where α takes respectively the values 0.25, 0.5, 1, 2, and 4)

To evaluate the validity of (13), we run LB-CMA-ES on the `bbob-mixint` testbed [13] with lower bound $\sigma_{\text{LB}} = \min\left(\alpha\frac{\mu_{\text{eff}}}{N}, 0.2\right)$ for $\alpha \in 0.25, 0.5, 1, 2, 4$.

Figure 2 shows some exemplary results on two functions in dimension 40. On the separable ellipsoid, more difficult target values are reached about twice as fast with $\alpha = 1$ as with the other tested values ($p \approx 10^{-4.5}$). We observe the same behavior on the 40D sphere function ($p \approx 10^{-3.5}$, not shown). On the linear slope in dimension 40, larger values for α are generally faster: with $\alpha = 1$, we reach the more difficult targets two times slower than with $\alpha \in 2, 4$ ($p \approx 10^{-3.5}$) but at least 3 times faster than with $\alpha \in 0.5, 0.25$ ($p \approx 0.02$). A lower bound which is two times larger than Eq. 13 but still clipped at 0.2 has mostly similar performance but is more at risk to disrupt convergence.

2.3 Integer Centering

When $\sigma^{(t)}_{\text{std(j)}}$ is small, a successful integer mutation will often have little effect because its value is likely to be close to the original integer value and because the impact of a single solution on the mean update diminishes with increasing population size. To mitigate these effects, we set successfully mutated values to the center of the integer interval. Furthermore, we aim to maintain the unweighted "unselected" average taken over the μ best sampled solutions.

Let $(.)_j$ denote the j-th coordinate of a vector, let $I_\mu = \{1{:}\lambda, \ldots, \mu{:}\lambda\}$ denote the index set of the μ best solutions from (3), and for $i \in I_\mu$ let $\boldsymbol{x}_i^{(t^-)} = \boldsymbol{x}_i^{(t)}$ from (2). Before applying Eq. (4), we set

$$(\boldsymbol{x}_i^{(t)})_j \leftarrow \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] \quad \text{if} \quad \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] \neq \text{int}\left[(\boldsymbol{m}^{(t)})_j\right] \ , \tag{20}$$

for $i \in I_\mu$ and $j = 1, \ldots, N$. That is, we apply integer centering to $(\boldsymbol{x}_i^{(t)})_j$ if it has a different integer value than the mean.

The centering of the candidate solutions introduces a bias,

$$b_j^{(t)} = \sum_{i \in I_\mu} \left((\boldsymbol{x}_i^{(t)})_j - (\boldsymbol{x}_i^{(t^-)})_j\right) \quad \text{for} j = 1, \ldots, N \ . \tag{21}$$

To compensate for this bias, we set for $i \in I_\mu$ and $j = 1, \ldots, N$

$$(\boldsymbol{x}_i^{(t)})_j \leftarrow (\boldsymbol{x}_i^{(t^-)})_j + \mathbb{1}(i,j)\alpha_j \Delta_{ij} \tag{22}$$

such that $\alpha_j \leq 1$ minimizes $b_j^{(t)}$, where $\Delta_{ij} = \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] - (\boldsymbol{x}_i^{(t^-)})_j$ and

$$\mathbb{1}(i,j) = \begin{cases} 1 & \text{if int}\left[(\boldsymbol{x}_i^{(t)})_j\right] = \text{int}\left[(\boldsymbol{m}^{(t)})_j\right] \text{ and } b_j^{(t)}\Delta_{ij} < 0 \\ 0 & \text{otherwise} \end{cases} \ . \tag{23}$$

That is, when $(\boldsymbol{x}_i^{(t)})_j$ has not been centered yet, we move it by α_j towards its centered version, given this move reduces the bias (21).

These modifications do not change the f-value of the solutions, that is, $f(\boldsymbol{x}_i^{(t)}) = f(\boldsymbol{x}_i^{(t^-)})$, hence they do not change (3). The $\boldsymbol{y}_i^{(t)}$ from (2) are reassigned according to (20) and (22).

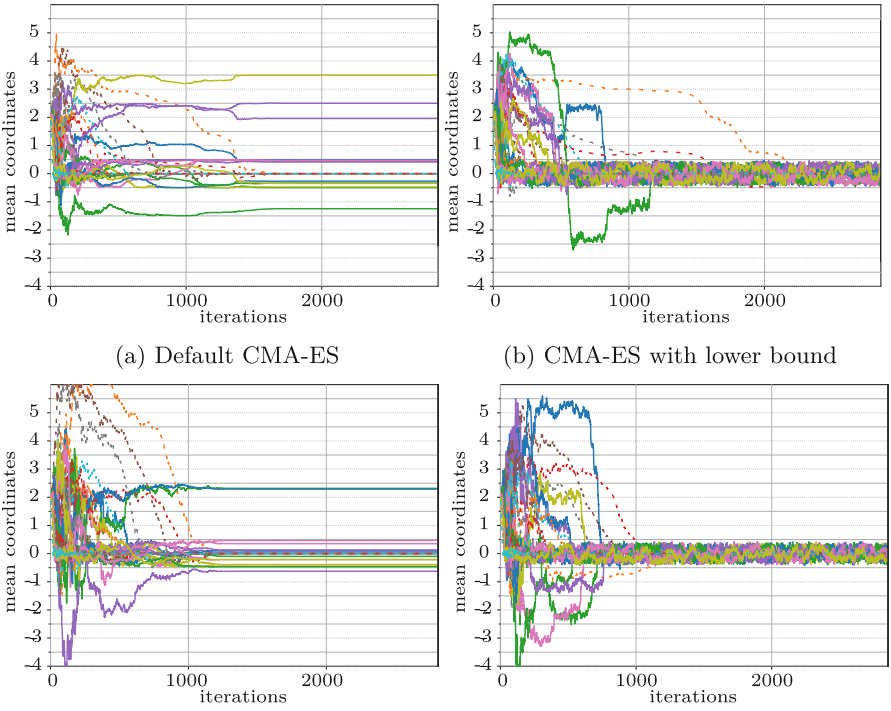
All our code is based on the cma Python package, commit 334abfc.

3 Single Runs of the Different Variants

To illustrate the effect of the above modifications, some optimization runs are shown in Fig. 3. All graphs result from the optimization of the 30-dimensional mixed-integer ellipsoid function (see caption). We desire the continuous variables to linearly converge to the optimum at zero and the integer variables to end up in $[-0.5, 0.5]$.

Without integer handling (Fig. 3a), integer variables get stuck after about 1400 iterations where four variables assume a nonoptimal value. Because the step size converges to zero, they will not change their final position anymore.

When the standard deviations are lower bounded according to Sect. 2.2, integer variables cannot get anymore stuck and end up in the optimal interval $[-0.5, 0.5]$ where their behavior resembles a bounded random walk (Fig. 3b). With integer centering only, variables appear to move more swiftly before iter-



(a) Default CMA-ES (b) CMA-ES with lower bound

(c) CMA-ES with integer centering (d) Lower bound and integer centering

Fig. 3. Evolution of the mean, $\boldsymbol{m}^{(t)}$, of CMA-ES with default population size $\lambda = 14$ on the 30-dimensional ellipsoid function $f_{\text{elli}}(x) = \sum_{i=1}^{N} 10^{6\frac{i-1}{N-1}} x_i^2$ where variables with even/odd index are continuous/integer and plotted in dashed/solid, respectively. Initial mean and step-size are $\boldsymbol{m}^{(0)} = (2, ..., 2)$ and $\sigma^{(0)} = 0.1$, respectively. Integer variable values in $[-0.5, 0.5]$ are optimal.

ation 1000, however, two integer variables still get stuck at a nonoptimal value (Fig. 3c). The effect of combining lower bound and integer centering appears to be additive (Fig. 3d): integer variables appear to move more swiftly, and all integer variables reach the optimal interval within about 800 iterations.

4 Performance Assessment

We assess the impact of lower bounding the standard deviations of integer variables and of integer centering on the 24 functions of the `bbob-mixint` testbed [13]. The functions are benchmarked in dimensions 5, 10, 20, 40 and 80 where 20% of these variables are continuous and the rest are integer, 20% with arity 2, 4, 8 and 16, respectively. The k-ary variables take values between 0 and $k-1$. For each function, 51 target f-values are defined as $f_{\text{opt}} + \Delta f$ where f_{opt} is the minimal function value and $\Delta f = 10^k$ with $k = 2, 1.8, \ldots, -8$. The number of function evaluations to reach each target is then recorded.

As the objective function remains constant for integer values smaller than 0 and larger than k, we bound their search domain to $[-1/2, k + 1/2]$. Continuous variables are not bounded. Boundaries are handled by adding a penalty term to the objective function [9] with the cma.BoundPenalty class. The initial coordinate-wise standard deviation is set to one fifth of the bounded range and to $10/5 = 2$ for continuous variables.

We benchmark all four combinations with and without LB and/or IC, referred to as base-CMA, LB-CMA, IC-CMA and LB+IC-CMA. All tested variants use IPOP-CMA-ES [2]: if a termination condition is met before the allowed budget is exhausted, we restart CMA-ES with doubled population size. The termination conditions `tolflatfitness` and `tolfunhist` are changed from their default values to 5 and 0, respectively.

We compare our variants with two previously benchmarked algorithms: CMA-ES-pycma, with some basic integer handling [13] and CMA-ES with margin (CMA-ESwM) which, similar to LB+IC-CMA, modifies coordinate-wise standard deviations and the mean vector to ensure a minimal marginal probability of exploring new integer variables [3,4].

Empirical runtime distributions [7] are shown by function groups in Fig. 4 and for each function in Figs. 5 and 6.[1] Mentioned p-values are computed within the COCO platform [8] and represent the result of a ranksum test.

First we investigate the effect of the lower bound. The LB-CMA algorithm takes at least 10 times less functions evaluations than base-CMA for solving the target $\Delta f = 10^{-7}$ on the separable ellipsoid $f2$ in dimension 10, 20, 40 and 80 ($p \approx 10^{-4.5}$). Also, LB-CMA reaches the target $\Delta f = 10^{-7}$ at least 5 times faster than base-CMA on the 80D functions `f1`, `f5`, `f12`, `f13` ($p \approx 0.02$ for `f12`, $p \approx 10^{-2.5}$ for `f13`, $p \approx 10^{-3.5}$ for `f5` and $p \approx 10^{-4.5}$ for `f1`). On the other hand base-CMA does not reach any target faster than LB-CMA. Introducing a lower

[1] All data and a code example can be found at https://trmarty.github.io/LB-IC-CMA-ES-data/.

bound on the standard deviations mostly improves the behavior of the algorithm on unimodal functions.

Next, we examine the effect of integer centering. The IC-CMA reaches the target $\Delta f = 10^{-7}$ at least twice as fast as base-CMA on the 80D functions `f2`, `f14`, `f18` ($p \approx 0.02$ for `f18`, $p \approx 10^{-2.5}$ for `f14` and $p \approx 10^{-4.5}$ for `f2`). However, base-CMA solves function `f15` eight times and function `f24` four times out of 15 whereas IC-CMA never succeeds.

With both lower bound *and* integer centering (LB+IC-CMA), the number of functions evaluations to reach the target 10^{-7} is reduced by a factor 4 compared to LB-CMA on the 80D functions `f14`, `f17` and `f18` ($p \approx 10^{-2.5}$ for `f18`, $p \approx 10^{-3.5}$ for `f17` and $p \approx 10^{-4.5}$ for `f14`). However, LB-CMA optimizes `f24` up to the final target while LB+IC-CMA does not even reach the easy target of $\Delta f = 10$. Overall, LB+IC-CMA has lower runtimes than the base-CMA in dimension 80 on functions `f1`, `f2`, `f12`, `f13`, `f14`, `f17` and `f19` at the expense of not solving the multimodal functions `f15` and `f24`.

Finally, the LB+IC-CMA-ES algorithm is compared with CMA-ES with margin. The target $\Delta f = 10^{-7}$ is reached faster with LB+IC-CMA than with CMA-ESwM on functions `f8` and `f24` in dimension 5, `f1`, `f2`, `f8`, `f12`, `f14`, `f17` in dimension 10, `f1`, `f2`, `f7`, `f12`, `f13`, `f15` and `f17` in dimension 20, `f1`, `f2`, `f7`, `f12` and `f13` in dimension 40 and on functions `f2`, `f12`, `f17` and `f18` in dimension 80 ($p < 0.05$ for all functions). On the other hand, CMA-ESwM is 2, 4 and 10 times faster on the linear slope `f5` in dimension 20, 40 and 80, respectively ($10^{-5} < p < 10^{-3}$) which has its optimum on the boundary of the domain. Removing the boundary handling speeds up LB+IC-CMA by a factor of three to hit the last target $\Delta f = 10^{-8}$ ($p \approx 10^{-5}$).

In all other cases, both algorithms have similar performance. In summary, LB+IC-CMA performs better than CMA-ESwM on 24 of 120 functions and worse on three.

Table 1. Final population size in dimension 40 and 80 for functions with at least one successful trial and one restart.

function id	min	med	max
f1, 80D	17	34	34
f2, 40D	15	15	30
f2, 80D	34	34	34
f5, 40D	15	15	30
f5, 80D	34	34	68
f7, 40D	120	240	960

function id	min	med	max
f12, 40D	15	15	30
f12, 80D	17	17	34
f13, 40D	15	15	30
f13, 80D	17	17	34
f14, 40D	15	15	60
f14, 80D	17	17	34

function id	min	med	max
f15, 40D	60	120	240
f17, 40D	15	30	30
f17, 80D	34	68	136
f18, 40D	15	30	120
f18, 80D	68	136	272
f21, 40D	15	30	480
f21, 80D	34	34	136

Table 1 shows final population sizes for functions with at least one successful run and *at least one restart* in dimensions 40 and 80 (11 functions). The default

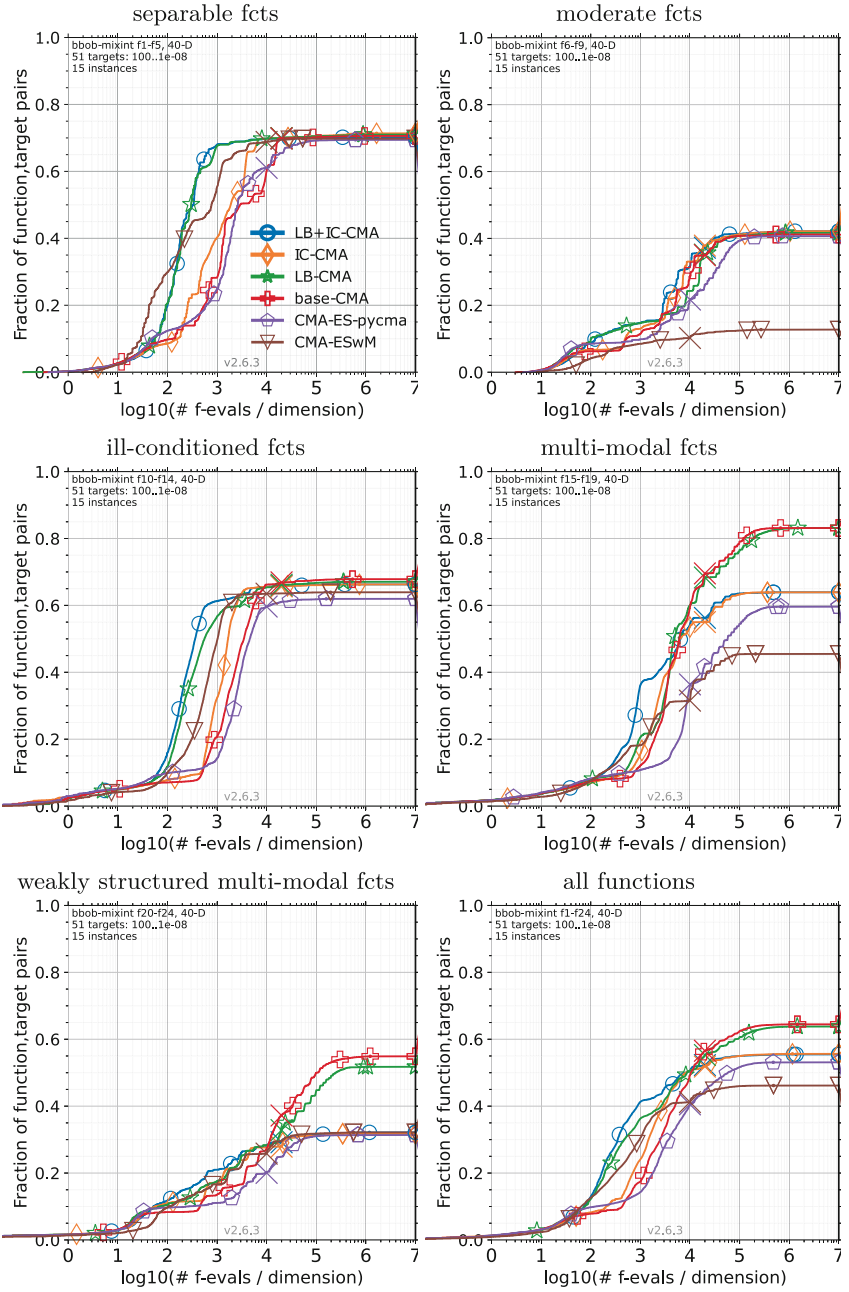


Fig. 4. Bootstrapped empirical cumulative distributions of the number of f-evaluations divided by dimension for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 40-D. The performance of CMA-ESwM on the moderate function group is negatively affected by missing data on `f9`.

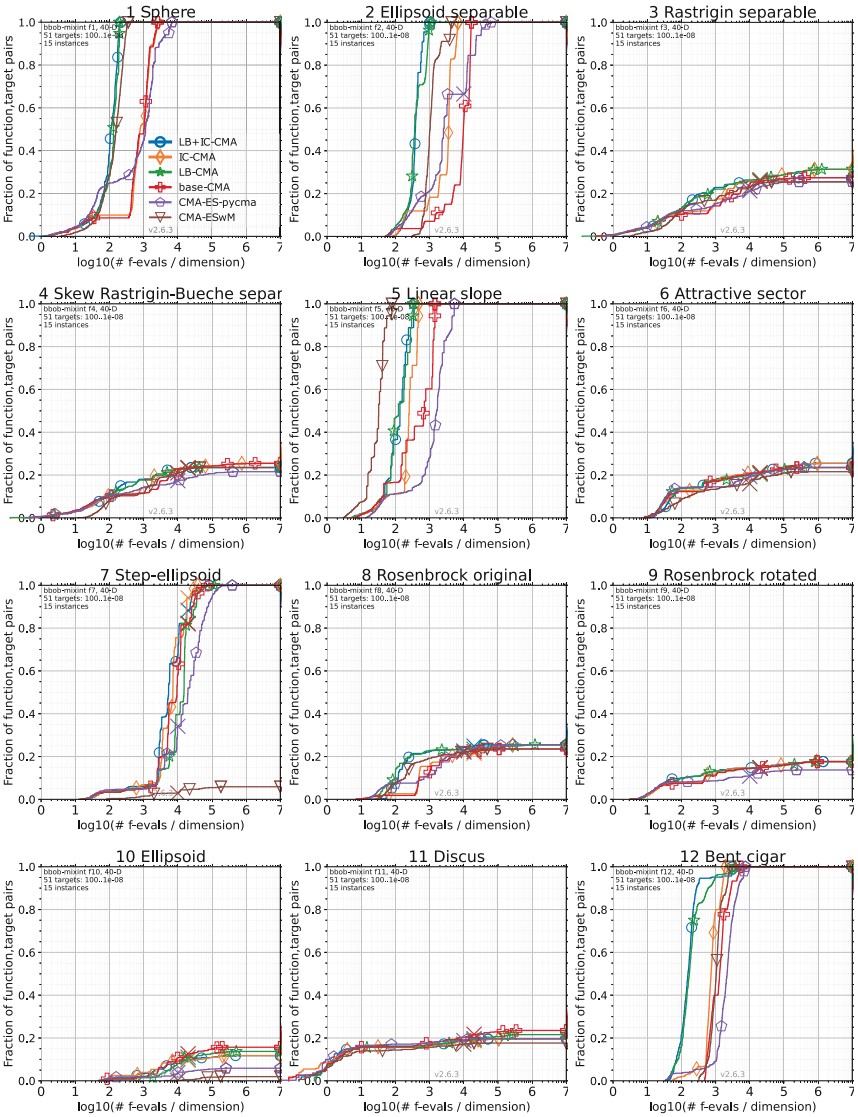


Fig. 5. Empirical cumulative distributions of simulated (bootstrapped) runtimes, measured in number of f-evaluations divided by dimension, for the 51 targets $10^{[-8..2]}$ in dimension 40.

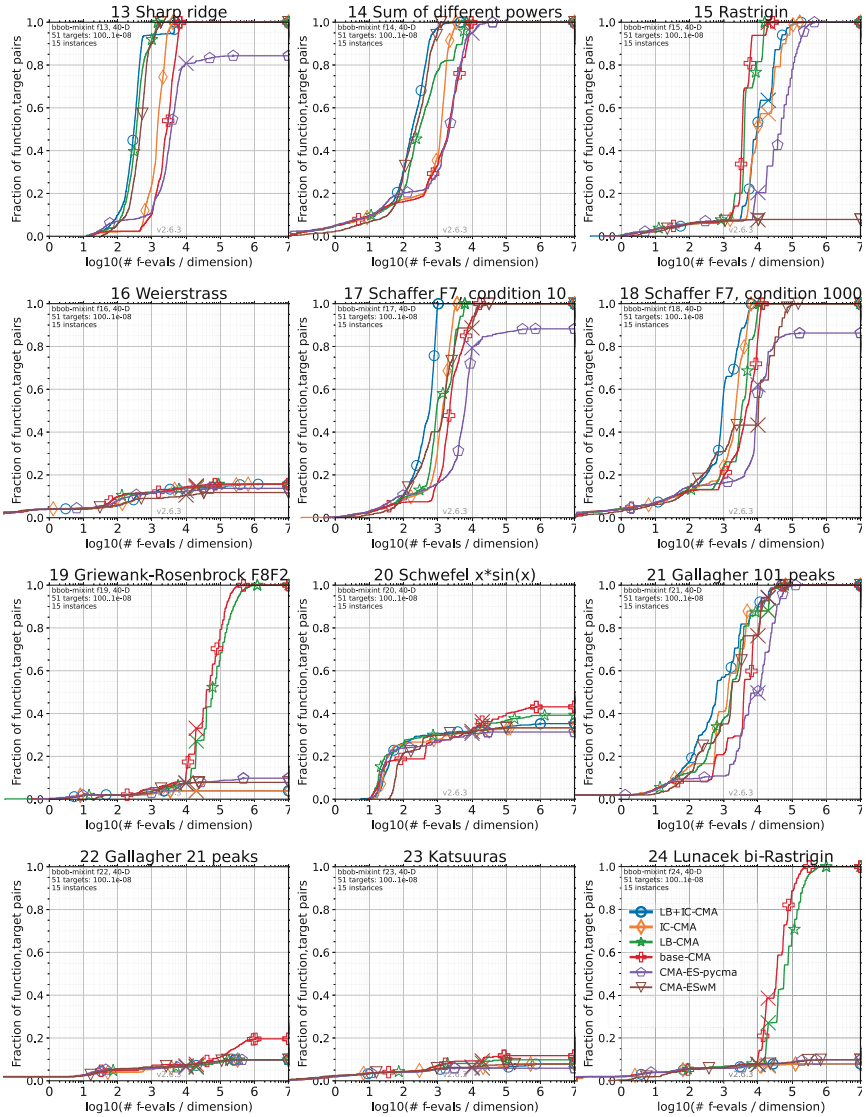
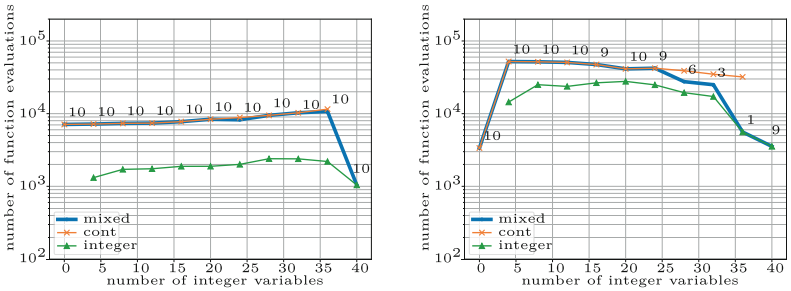


Fig. 6. Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations divided by dimension, for the 51 targets $10^{[-8..2]}$ in dimension 40.

population size is 15 and 17 in dimension 40 and 80, respectively, and with IPOP each restart doubles the population. Hence, the final population size also implies the number of conducted restarts. The largest population sizes can be observed on the step ellipsoid function `f7` (up to 960). In dimension 80, LB+IC-CMA often restarts even on the sphere function `f1`. Apparently, the `tolfun` termination condition triggers too early in larger dimension.



(a) Sphere (b) Ellipsoid

Fig. 7. Average number of function evaluations to solve the 40D sphere and ellipsoid function up to $f = 10^{-10}$ with LB+IC-CMA, plotted versus the number of integer variables. Results are for the mean $\boldsymbol{m}^{(t)}$ in the continuous (orange) and the integer (green) subspace and for the best solution on the overall mixed-integer problem (blue). The number of successful runs out of 10 is given too. (Color figure online)

Finally, Fig. 7 shows average runtimes versus the percentage of integer variables on the sphere function and the ellipsoid function (see Fig. 3) with a conditioning of 10^6 in dimension 40. A run is successful if the best solution reaches the target value of 10^{-10}. On the sphere integer function (Fig. 7a), problems with at least one continuous variable are solved within 7000 to 11 000 evaluations. Adding only few integer variables to this continuous problem does not increase the evaluation time by much ($\times 1.1$ between 0 and 4 integer variables), however adding a few continuous variables to an integer-only problem has a much larger impact ($\times 10$ between 40 and 36 integer variables). The ellipsoid function (Fig. 7b) is harder to solve. With increasing number of integer variables, the success rate falls below 100% and with 36 integer variables, the success rate drops to 10%. The integer-only and continuous-only problems takes around 3500 evaluations to find the optimum which is approximately 10 times faster than all other mixed problems that are solved with a probability larger than 50%. With 20% continuous variables, the BBOB benchmark represents comparatively difficult problems in terms of fraction of integer variables.

5 Summary and Conclusion

We propose two comparatively simple modifications of CMA-ES to improve the performance of the algorithm on problems with (some) integer variables. First,

we derived a lower bound for the standard deviation of integer coordinates which depends on the dimension of the problem and the population size. The bound aims to be low enough to preserve a large enough effective population size in the continuous subspace. Second, we move successfully mutated integer variables to the center of their integer interval to adopt good mutations of integer variables more swiftly.

The lower bound greatly reduces the runtime of CMA-ES on functions `f1`, `f2`, `f5`, `f12` and `f13` of the `bbob-mixint` suite [13]. Additional integer centering leads to faster convergence on functions `f14`, `f17` and `f18`, however also to a performance degradation on the multimodal functions `f19` and `f24` and, in higher dimension, on `f15`. The cause of this degradation remains unclear. We suspect that restarts on `f1` and `f7` lead to unnecessary performance impediments too. The LB+IC-CMA-ES compares overall favorably to the CMA-ES with margin [4], especially on functions `f2` and `f12` in dimension 10–80 and on eight other functions depending on the dimension.

Many BBOB mixed-integer multimodal functions remain unsolved by the current algorithms. The proposed variant, LB+IC-CMA-ES, however improves over previous versions and is a comparatively simple new baseline to compare with.

References

1. Akimoto, Y., Hansen, N.: Diagonal acceleration for covariance matrix adaptation evolution strategies. Evol. Comput. **28**(3), 405–435 (2020)
2. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776. IEEE (2005)
3. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: Benchmarking CMA-ES with margin on the bbob-mixint testbed. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1708–1716 (2022)
4. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 639–647 (2022)
5. Hansen, N.: A CMA-ES for mixed-integer nonlinear optimization. INRIA research report, RR-7751 (2011)
6. Hansen, N.: The CMA evolution strategy: a tutorial. arXiv preprint arXiv:1604.00772 (2016)
7. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**(1), 114–144 (2021)
8. Hansen, N., et al.: COmparing Continuous Optimizers: numbbo/COCO on Github (2019). https://doi.org/10.5281/zenodo.2594848
9. Hansen, N., Niederberger, A.S., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Trans. Evol. Comput. **13**(1), 180–197 (2008)

10. Ikeda, K., Ono, I.: Natural evolution strategy for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 831–838 (2023)
11. Li, R., et al.: Mixed integer evolution strategies for parameter optimization. Evol. Comput. **21**(1), 29–64 (2013)
12. Marty, T., Semet, Y., Auger, A., Héron, S., Hansen, N.: Benchmarking CMA-ES with basic integer handling on a mixed-integer test problem suite. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, pp. 1628–1635 (2023)
13. Tušar, T., Brockhoff, D., Hansen, N.: Mixed-integer benchmark problems for single- and bi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 718–726 (2019)
14. Watanabe, Y., Uchida, K., Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: (1+1)-CMA-ES with margin for discrete and mixed-integer problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 882–890 (2023)