A Covariance Matrix Self-Adaptation Evolution Strategy for Optimization under Linear Constraints

Patrick Spettel, Hans-Georg Beyer, and Michael Hellwig

***Abstract*—This paper addresses the development of a covariance matrix self-adaptation evolution strategy (CMSA-ES) for solving optimization problems with linear constraints. The proposed algorithm is referred to as Linear Constraint CMSA-ES (lcCMSA-ES). It uses a specially built mutation operator together with repair by projection to satisfy the constraints. The lcCMSA-ES evolves itself on a linear manifold defined by the constraints. The objective function is only evaluated at feasible search points (interior point method). This is a property often required in application domains such as simulation optimization and finite element methods. The algorithm is tested on a variety of different test problems revealing considerable results.**

***Index Terms*—Constrained Optimization, Covariance Matrix Self-Adaptation Evolution Strategy, Black-Box Optimization Benchmarking, Interior Point Optimization Method**

I. Introduction

THE Covariance Matrix Self-Adaptation Evolution Strategy (CMSA-ES) [1] variant called Constraint CMSA-ES (cCMSA-ES) was proposed in [2]. It showed promising results in portfolio optimization applications. Therefore, further research on ES design principles for constrained optimization problems is of interest. The CMSA-ES and linear constraints are chosen as a first step. This is because the CMSA-ES is arguably one of the most simple variants of Covariance Matrix Adaptation (CMA) ESs [3], [4]. In addition, linear constraints are the most simple constraints after box constraints. But this does not mean that it is only of theoretical interest. Such problems occur in practical applications. Examples include risk management in finance [5], [6], [7], agriculture [8], hybrid dynamic systems [9], model predictive control [10], controlled perturbation for tabular data [11], and optimization of heat exchanger networks [12]. Further, the CEC 2011 real world optimization problem competition contains an electrical transmission pricing problem based on the IEEE 30 bus system [13, Prob. 9 in Sec. 8]. Another example is a problem from the area of chemistry. The chemical composition of a complex mixture under chemical equilibrium conditions has to be determined. This problem is described in detail in [14, pp. 47-49].

Evolutionary Algorithms (EAs) in general are well-suited for scenarios in which objective function and/or constraint functions cannot be expressed in terms of (exact) mathematical expressions. Moreover, if that information is incomplete or if that information is hidden in a black-box, EAs are a good choice as well. Such methods are commonly referred to as direct search, derivative-free, or zeroth-order methods [15], [16], [17], [18]. In fact, the unconstrained case has been studied well. In addition, there is a wealth of proposals in the field of Evolutionary Computation dealing with constraints in real-parameter optimization, see e.g. [19]. This field is mainly dominated by Particle Swarm Optimization (PSO) algorithms and Differential Evolution (DE) [20], [21], [22]. For the case of constrained discrete optimization, it has been shown that turning constrained optimization problems into multi-objective optimization problems can achieve better performance than the single-objective variant with a penalty approach for some constrained combinatorial optimization problems, e.g., [23], [24], [25].

ESs for constrained optimization have not yet been studied extensively. Early work includes the $(1 + 1)$-ES for the axis-aligned corridor model [26], the $(1, \lambda)$-ES for the same environment [27], and the $(1+1)$-ES for a constrained, discus-like function [28]. Moreover, a stochastic ranking approach was proposed in [29]. An ES for constrained optimization was proposed in [30]. For the CMA-ESs, in addition to the cCMSA-ES [2], a $(1 + 1)$-CMA-ES based on active covariance matrix adaptation is presented in [31]. There exists an extension of this idea to a (μ, λ)-CMA-ES motivated by an application in the area of rocket design [32]. In [33] an Active-Set ES that is able to handle constraints is described. Further, an ES with augmented Lagrangian constraint handling is presented in [34]. The cCMSA-ES [2] uses two mechanisms to ensure the feasibility with respect to box and equality constraints. First, mutations are generated in such a way that they lie on the hypersurface defined by the equality constraint. Second, a repair mechanism is applied to offspring violating the box-constraints.

Being based on these ideas, the contribution of this work is a theoretically motivated and principled algorithm design. The proposed algorithm is an interior point ES. It is able to optimize a black-box objective function subject to general linear constraints. The peculiarity of this design is that the algorithm treats the function f to be optimized as a black-box. However, only *feasible* candidate solutions are used to query the black-box f. This is in contrast to most of the evolutionary algorithms proposed for constrained black-box optimization. However, it is a property often required in

Manuscript received Month xx, xxxx; revised Month xx, xxxx; accepted Month xx, xxxx. Date of publication Month xx, xxxx; date of current version Month xx, xxxx. The authors thank Asma Atamna for providing the simulation code for the ES with augmented Lagrangian constraint handling. This work was supported by the Austrian Science Fund FWF under grant P29651-N32.

The authors are with the Research Center Process and Product Engineering at the Vorarlberg University of Applied Sciences, Dornbirn, Austria (e-mail: Patrick.Spettel@fhv.at; Hans-Georg.Beyer@fhv.at; Michael.Hellwig@fhv.at).

Digital Object Identifier xx.xxxx/TEVC.xxxx.xxxxxxx

the field of simulation optimization, e.g. in Computational Fluid Dynamics (CFD) optimizations. In CFD optimizations, constraint violations on simulator input parameters may result in simulator crashes.[1] In [35], concrete real-world examples are provided for different constraint types. Among those examples, a ground water optimization problem [36] is provided for which (some) constraints are not allowed to be violated. Because the simulator only supports extraction but not injection, the lower bounds on the pumping rate values must hold for the simulation. Physical requirements like that usually prohibit the violation of (some) constraints. Further, it is an important property for problems that cannot tolerate even small infeasibility rates. Such problems can be a topic in finance or business applications, i.e., the optimization of a function subject to a constant amount of total money in the system. Moreover, the mutation operator and the repair method are specially designed. The ES moves completely on a linear manifold defined by the constraints. For this design, the theory is an essential part.

The rest of the paper is organized as follows. In Sec. II the optimization problem is presented. Then, the proposed algorithm is described in Sec. III and simulation results are presented in Sec. IV. Finally, Sec. V summarizes the main results and provides an outlook.

Notations Boldface $\mathbf{x} \in \mathbb{R}^D$ is a column vector with D real-valued components. $\mathbf{x}^T$ is its transpose. x_d and equivalently $(\mathbf{x})_d$ denote the d-th element of a vector $\mathbf{x}$. $x_{(k:D)}$ and equivalently $(\mathbf{x})_{(k:D)}$ are the order statistic notations, i.e., they denote the k-th smallest of the D elements of the vector $\mathbf{x}$. $||\mathbf{x}|| = \sqrt{\sum_{d=1}^{D} x_d{}^2}$ denotes the euclidean norm (ℓ_2 norm) and $||\mathbf{x}||_1 = \sum_{d=1}^{D} |x_d|$ the ℓ_1 norm. $\mathbf{X}$ is a matrix, $\mathbf{X}^T$ its transpose, and $\mathbf{X}^+$ its pseudoinverse. $\mathbf{0}$ is the vector or matrix (depending on the context) with all elements equal to zero. $\mathbf{I}$ is the identity matrix. $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ denotes the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$. $\mathcal{N}(\mu, \sigma^2)$ is written for the normal distribution with mean μ and variance σ^2. $\mathcal{U}[a, b]$ represents the continuous uniform distribution with lower bound a and upper bound b. The symbol $\sim$ means "distributed according to", $\gg$ "much greater than", and $\simeq$ "asymptotically equal". A superscript $\mathbf{x}^{(g)}$ stands for the element in the g-th generation. $\langle \mathbf{x} \rangle$ denotes the mean (also centroid) of a parameter of the μ best individuals of a population, e.g. $\langle \tilde{\mathbf{z}} \rangle = \frac{1}{\mu} \sum_{m=1}^{\mu} \tilde{\mathbf{z}}_{(m:\lambda)}$.

II. Optimization Problem

We consider the (non-linear) optimization problem

$$f(\mathbf{x}) \to \min! \tag{1a}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \tag{1b}$$

$$\mathbf{x} \geq \mathbf{0} \tag{1c}$$

where $f : \mathbb{R}^D \to \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{K \times D}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^K$. Note that Eqs. (1b) and (1c) form a linear constraint system in standard form. Any linear inequality constraints and bounds can be transformed into an equivalent problem of this form. In particular, a vector satisfying the constraints in the transformed system, also satisfies the constraints in the original system. A method for this transformation is presented in the supplementary material (Sec. VI-B).

[1]Note that although interior point methods only run the simulator with feasible input parameters, problems can still occur. Feasible input parameters can lead to crashes because of possible parameter combinations that were never thought of. And constraints are often defined on simulator outputs. Such cases need a different treatment not subject of this paper.

III. Algorithm

Based on the Covariance Matrix Self-Adaptation Evolution Strategy (CMSA-ES) [1], we propose an algorithm for dealing with problem (1). In particular, we describe the design of the linear constraint $(\mu/\mu_I, \lambda)$-CMSA-ES (lcCMSA-ES).

The $(\mu/\mu_I, \lambda)$-CMSA-ES makes use of σ-self-adaptation and a simplified covariance update. It starts from a parental individual $\mathbf{x}$ (line 5 in Alg. 3 corresponds to this step) and an initial mutation strength σ (part of line 2 in Alg. 3). The generational loop consists of two main steps. First, λ offspring are generated. For every offspring l, its mutation strength $\tilde{\sigma}_l$ is sampled from a log-normal distribution. Then, the offspring's parameter vector is sampled from a multi-variate normal distribution $\mathcal{N}(\mathbf{x}, \tilde{\sigma}_l \mathbf{C})$ (lines 15 to 28 in Alg. 3 is the offspring generation (extended for the constrained case)). Second, the parental individual is updated for the next generation. For this, the parameter vectors and the mutation strengths of the best μ offspring are averaged. Then, the covariance is updated (lines 29 to 34 in Alg. 3). This completes the steps for one iteration of the inner loop.

The method proposed in this paper represents an interior point method, i.e., the individuals evaluated in Eq. (1a) are always feasible. In other words, the ES moves inside the feasible region while searching for the optimum. Concretely, this is realized by starting off with an initial centroid that is feasible (Sec. III-A). Mutation (Sec. III-B) is performed in the null space of $\mathbf{A}$ in order to keep the mutated individuals feasible with respect to Eq. (1b). It is possible that after mutation Eq. (1c) is violated. Repair (Sec. III-C) by projection to the positive orthant is performed in such cases.

A. Initialization of the Initial Centroid

The problem for the initialization of the initial parental centroid is to find an $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. For the first part, the system of linear equations can be solved. This solution $\mathbf{x}$ possibly violates the second part. In that case the repair approach of the ES (Sec. III-C) is applied to this initial solution. Under the assumption that the linear system solver and the repair operator are deterministic, this whole initialization is deterministic. This means that for the same $\mathbf{A}$ the same $\mathbf{x}$ is computed every time. In order to use the algorithm in a restarted fashion, random initialization is important. For this, an initial random movement of $\mathbf{x}$ can be obtained in a similar way as for the mutation (Sec. III-B).

B. Mutation

The goal of mutation is to introduce variation to the population of candidate solutions. In the unconstrained case one could simply add a random vector to the parental centroid.

But the constraints make this more complicated. Since the proposed method is an interior point method, mutated individuals that violate the constraints must be repaired. One option would be to design a mutation operator that does not violate any constraints. Here, the approach is a mixture of both. The mutation operator does not violate the linear equality constraints but it does possibly violate the non-negativity constraint. The latter case is handled through repair by projection. For the former note that $\mathbf{A}(\mathbf{x}_{\text{inh}} + \mathbf{x}_{\text{h}}) = \mathbf{b}$ where $\mathbf{x}_{\text{inh}}$ is an inhomogeneous and $\mathbf{x}_{\text{h}}$ is a homogeneous solution. Thus, $\mathbf{x}_{\text{h}} \in \text{null}(\mathbf{A})$ and therefore $\mathbf{A}\mathbf{x}_{\text{h}} = \mathbf{0}$. Let N be the dimension and $\mathbf{B} \in \mathbb{R}^{D\times N}$ an orthonormal basis of the null space $\text{null}(\mathbf{A})$, i.e., $\mathbf{B}^T\mathbf{B} = \mathbf{I}$ and $\mathbf{A}\mathbf{B} = \mathbf{0}$ hold. Mutations are performed in $\text{null}(\mathbf{A})$ and therefore do not violate Eq. (1b). This means that a mutation vector $\mathbf{s}$ in the null space is sampled from a normal distribution with zero mean and the covariance matrix $\mathbf{C}$, i.e., $\mathbf{s} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^{N\times N})$. Transforming this $\mathbf{s}$ into the parameter space and scaling it with the mutation strength σ yields a mutation vector $\mathbf{z} = \sigma\mathbf{B}\mathbf{s}$ in the parameter space. This $\mathbf{z}$ can be added to the parental centroid (or the initial centroid for initial value randomization)

$$\mathbf{x}^{(g+1)} = \mathbf{x}^{(g)} + \mathbf{z} = \mathbf{x}^{(g)} + \sigma\mathbf{B}\mathbf{s}. \tag{2}$$

Assuming the parental centroid satisfies the linear constraints $\mathbf{A}\mathbf{x}^{(g)} = \mathbf{b}$, a short calculation shows that the mutated offspring fulfills them as well:

$$\begin{aligned}\mathbf{A}\mathbf{x}^{(g+1)} &= \mathbf{A}(\mathbf{x}^{(g)} + \sigma\mathbf{B}\mathbf{s}) = \mathbf{A}\mathbf{x}^{(g)} + \mathbf{A}(\sigma\mathbf{B}\mathbf{s}) \\ &= \mathbf{A}\mathbf{x}^{(g)} + \sigma\underbrace{(\mathbf{A}\mathbf{B})}_{\mathbf{0}^{K\times N}}\mathbf{s} = \mathbf{b} + \mathbf{0}^{K\times 1} = \mathbf{b}.\end{aligned} \tag{3}$$

C. Repair

Eq. (1b) is not violated through mutation. But violation of Eq. (1c) must be dealt with. The approach followed here is repair by projection onto the positive orthant.

Although repair by minimal change is intuitively the most plausible approach, it is worth noting that the repair in the ES does not have to be optimal. It is enough to find a point on the positive orthant that is approximately at minimal distance to the infeasible point and the evolution strategy is still able to move. Regarding the minimal distance, there come different distance definitions into mind, e.g. the ℓ_2 and the ℓ_1 norm, respectively. We use the latter instead of the squared euclidean distance (squared ℓ_2 norm). In addition, we propose another projection method based on random reference points.

The projection formulated as the minimization of the squared ℓ_2 norm leads to a Quadratic Program (QP). This is, however, computationally expensive. In particular, if the evolution strategy moves near the boundary of the feasible region the probability of repair is high. For this reason minimizing the ℓ_1 distance for repair and a new projection approach based on random reference points are investigated with the goal of having a projection method with a faster asymptotic runtime. The runtime scaling behavior of the different projection approaches has been experimentally compared. The comparison plot is provided in the supplementary material (Fig. 4). Additionally, the projection quality has been experimentally compared for the different projection approaches (see Fig. 9).

Algorithm 1 Initialization of the set P of reference points for the Iterative Projection.

```
function initReferencePointsForIterativeProjection(x ∈ ℝ^D, numberOfPoints, A, b)
    B^{D×N} ← orthonormalize(null(A))
    for k ← 1 to numberOfPoints do
        p_k ← projectToPositiveOrthant(U[−||x||, ||x||], A, b)
    end for
    return({p_k | k ∈ {1, ..., numberOfPoints}})
end function
```

1) Projection by minimizing the ℓ_1 norm: Projection by minimizing the ℓ_1 norm results in a Linear Program (LP) and thus an LP solver can be used. The optimization problem is

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}'} \|\mathbf{x}' - \mathbf{x}\|_1 = \arg\min_{\mathbf{x}'} \left(\sum_k |x'_k - x_k|\right) \tag{4}$$

where $\mathbf{x}$ is the individual to be repaired. We introduce the convenience function

$$\hat{\mathbf{x}} = \text{projectToPositiveOrthant}(\mathbf{x}, \mathbf{A}, \mathbf{b}) \tag{5}$$

returning the solution $\hat{\mathbf{x}}$ of the problem (4). Technically, problem (4) can be turned into an LP

$$\begin{aligned} &\mathbf{1}^T\mathbf{z} \to \min! \\ \text{s.t. } &\mathbf{z} - \mathbf{x}' \geq -\mathbf{x} \\ &\mathbf{z} + \mathbf{x}' \geq \mathbf{x} \\ &\mathbf{A}\mathbf{x}' = \mathbf{b} \\ &\mathbf{x}' \geq \mathbf{0}. \end{aligned} \tag{6}$$

The introduced vector $\mathbf{z}$ is used to deal with the cases of the absolute value operator in (4). If $x'_k - x_k > 0$, then $-x'_k + x_k < 0$, if $x'_k - x_k < 0$, then $-x'_k + x_k > 0$ and if $x'_k - x_k = 0$, then $-x'_k + x_k = 0$. Consequently, the absolute value operator, the linear constraints, and the non-negativity constraint are handled. Depending on the format the LP solver expects as input, additional slack variable vectors can be introduced to turn it into an LP in standard form.

2) Projection based on random reference points: We propose a further alternative projection idea, the "Iterative Projection". The main idea is to create a set of one or more points

$$P = \{\mathbf{p}_k | k \in \{1, \ldots, \#\text{points}\}, \mathbf{A}\mathbf{p}_k = \mathbf{b}, \mathbf{p}_k \geq \mathbf{0}\} \tag{7}$$

inside the feasible region once in the beginning. These points are computed as follows. For each one a random point in the null space is chosen. It is then projected by the ℓ_1 minimization approach to get $\mathbf{p}_k$ fulfilling the constraints (Alg. 1 shows the pseudo-code). With this pre-processing in mind, we now consider a point $\mathbf{x}$ that needs to be repaired. For this point, movement in the null space towards a randomly chosen $\mathbf{p} \in P$ is possible without violating the linear constraints. As soon as the positive orthant is reached, the point is considered repaired. Intuitively, the points in P should be "far" inside the feasible region. This results in a movement that yields *different points on the boundary* for *different points outside the positive*

orthant. In other words, the positive orthant should be reached before getting "too close" to $\mathbf{p}$. More formally, let $\mathbf{d} = \mathbf{p} - \mathbf{x}$ be the direction of the movement towards $\mathbf{p}$. If the movement's starting point $\mathbf{x}$ fulfills $\mathbf{A}\mathbf{x} = \mathbf{b}$ movement in the null space to the positive orthant is possible without violating the linear constraints. Since all the negative elements should be zero, a factor α is necessary to compute

$$\mathbf{x}_{\text{projected}} = \mathbf{x} + \alpha\mathbf{d} \tag{8}$$

such that

$$\mathbf{x}_{\text{projected}} \geq \mathbf{0}. \tag{9}$$

The projection $\mathbf{x}_{\text{projected}}$ fulfills the linear equality constraints

$$\begin{aligned}\mathbf{A}\mathbf{x}_{\text{projected}} &= \mathbf{A}(\mathbf{x} + \alpha\mathbf{d}) = \mathbf{A}\mathbf{x} + \alpha\mathbf{A}(\mathbf{p} - \mathbf{x}) \\ &= \mathbf{b} + \alpha(\mathbf{b} - \mathbf{b}) = \mathbf{b}.\end{aligned} \tag{10}$$

Note that the target point $\mathbf{p} \in P$ is located in the positive orthant and satisfies the linear equality constraints $\mathbf{A}\mathbf{p} = \mathbf{b}$. Consequently, there exists an α that fulfills[2] Eqs. (8) and (9). One way would be to approach the positive orthant iteratively in the direction of $\mathbf{d}$ with a small α. But note that the α can also be computed such that all the negative elements are non-negative after the projection. The idea is to move towards the chosen reference point with an α that yields 0 for the component with the largest deviation from 0. This leads to an algorithm with a running time that is linear in the dimension of the vector. Alg. 2 shows the pseudo-code. The input is an $\mathbf{x}$ that needs to be projected and the linear constraint system $\mathbf{A}$ and $\mathbf{b}$ (Line 1). The precondition is checked by the assertion in Line 2. The result $\mathbf{x}_{\text{projected}}$ is initialized with $\mathbf{x}$ (Line 3) and the direction vector $\mathbf{d}$ is computed (Line 5) using a $\mathbf{p} \in P$ that is chosen randomly according to a uniform distribution (Line 4). Then, the worst alpha is computed in Lines 6 to 11. After the loop, the final projected vector is computed in Line 12 using the calculated α. The loop requires D steps. Every statement inside the loop can be implemented in constant time. This leads to a running time of $O(D)$.

Algorithm 2 Iterative Projection (runtime $O(D)$).

1: **function** projectToPositiveOrthantIter($\mathbf{x} \in \mathbb{R}^D, \mathbf{A}, \mathbf{b}, P$)
2: **assert**($D > 0 \wedge \mathbf{A}\mathbf{x} = \mathbf{b}$)
3: $\mathbf{x}_{\text{projected}} \leftarrow \mathbf{x}$
4: Choose a $\mathbf{p}$ uniformly at random from P
5: $\mathbf{d} \leftarrow \mathbf{p} - \mathbf{x}$
6: $\alpha \leftarrow 0$
7: **for** $k \leftarrow 1$ **to** D **do**
8: **if** $(\mathbf{x}_{\text{projected}})_k < 0 \wedge |(\mathbf{d})_k| > 0$ **then**
9: $\alpha \leftarrow \max(\alpha, -\frac{(\mathbf{x})_k}{(\mathbf{d})_k})$
10: **end if**
11: **end for**
12: $\mathbf{x}_{\text{projected}} \leftarrow \mathbf{x}_{\text{projected}} + \alpha\mathbf{d}$
13: **return**($\mathbf{x}_{\text{projected}}$)
14: **end function**

Experimental results for the different projection methods regarding runtime and projection quality are provided in the supplementary material (Sec. VI-E1). Fig. 4 shows the scaling behavior of the different projection methods. Fig. 9 shows the quality of the different projection methods. For this, Alg. 3 was configured with the different projection methods and run on different test problems.

[2]With $\alpha = 1$ we get $\mathbf{x}_{\text{projected}} = \mathbf{x} + (\mathbf{p} - \mathbf{x}) = \mathbf{p}$. And by construction we know that $\mathbf{A}\mathbf{p} = \mathbf{b}$ and $\mathbf{p} \geq 0$.

D. lcCMSA-ES Pseudo-Code

Alg. 3 shows the lcCMSA-ES in pseudo-code for the optimization problem described in Sec. II. It makes use of the ideas described in Secs. III-A to III-C.

An individual is represented as a tuple $\mathfrak{a}$. It consists of the objective function value $f(\mathbf{x})$, the parameter vector $\mathbf{x}$ for achieving this function value $f(\mathbf{x})$, the null space mutation vector $\mathbf{s}$, the mutation vector $\mathbf{z}$ and the mutation strength σ. The best-so-far (bsf) individual is tracked in $\mathfrak{a}_{\text{bsf}}$ and the corresponding generation in g_{bsf} (lines 11, 27, and 32 of Alg. 3).

In line 2 of Alg. 3 all the necessary parameters are initialized. The covariance matrix $\mathbf{C}$ is initialized to the identity matrix with dimension of the null space N (line 4). The initial solution is found by solving the linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ for an $\mathbf{x}_{\text{inh}}$ in line 5. It is then randomized in line 7 as described in Sec. III-A. In case the non-negativity constraint (Eq. (1c)) is violated, the initial solution is repaired (lines 8 to 10). Next, the generation loop is entered in line 13.

In every generation λ offspring are created (lines 15 to 28). The offspring's mutation strength is sampled from a log-normal distribution (line 16). The mutation direction in the null space is sampled from a normal distribution with the learned covariance and zero mean (line 17). Transformation of this mutation direction in the null space into the problem space yields the mutation direction in the problem space (line 18). Using this, the new offspring solution is calculated in line 19. It is repaired if it violates the non-negativity constraint. This is done by projection. The projection yields a new solution (line 21). From this, the mutation vector and the mutation vector in the null space are calculated back (lines 22 and 23).

The offspring are ranked according to the order relation "$\succ$" (line 29) to update the values $\mathbf{x}$, σ, and $\mathbf{C}$. They are updated with the mean values (denoted by $\langle\cdot\rangle$) of the corresponding quantities of the μ best individuals in lines 30, 33, and 34.

Since the goal is to minimize f and $f(\mathbf{x})$ is stored in the individual, the order relation is defined as

$$\mathfrak{a}_l \succ \mathfrak{a}_m \Leftrightarrow f(\tilde{\mathbf{x}}_l) < f(\tilde{\mathbf{x}}_m). \tag{11}$$

There are multiple termination criteria (line 36). The generation loop is terminated if a maximum number of generations is reached or the σ value falls below a threshold. In addition, the loop is stopped if the absolute or relative difference of $\mathbf{x}^{(g)}$ and $\mathbf{x}^{(g-G)}$ is below the threshold ε_{abs} or ε_{rel}, respectively. Further, if the best-so-far individual has not been updated for the last G_{lag} generations, the generational loop is quit.

The runtime of the generational loop of Alg. 3 is mainly dominated by two computation steps. The first expensive step is the eigendecomposition in the computation of $(\sqrt{\mathbf{C}})_{\text{normalized}}$. Second, the offspring generation step can be bounded as $O(\lambda \cdot t_{\text{proj}})$. This represents the worst case assuming

every generated offspring has to be repaired. The runtime cost of the repair step is denoted as t_{proj}. Consequently, assuming $\lambda = O(D)$, the projection starts to matter if t_{proj} gets about asymptotically quadratic in D.

Details concerning the covariance matrix $\mathbf{C}$ are explained in the following two subsections.

1) Computation of $\sqrt{\mathbf{C}}$: The covariance matrix $\mathbf{C}$ is symmetric and positive definite. Therefore, it holds that $\mathbf{C} = \mathbf{M}\mathbf{M}^T$ where $\mathbf{M} = \sqrt{\mathbf{C}}$. The computation of $(\sqrt{\mathbf{C}})_{\text{normalized}}$ for line 14 can be done every $\lfloor \tau_c \rfloor$-th generation to save time. This is possible because the changes to the covariance matrix are small in between these generations. Alg. 4 outlines the $\sqrt{\mathbf{C}}$ calculation steps. Note that $\det(\mathbf{M}_r)^{-\frac{1}{N}}$ is a normalization factor such that the determinant of $(\sqrt{\mathbf{C}})_{\text{normalized}}$ is one. The idea behind this is that the resulting transformation is volume-preserving.

2) Regularization of $\mathbf{C}$ for Computing $\sqrt{\mathbf{C}}$: When the strategy approaches the boundary, the selected (and repaired) mutation steps toward the boundary decrease rapidly. But the other directions are not affected. Consequently, the condition number of $\mathbf{C}$ increases rapidly.

Therefore, regularization of $\mathbf{C}$ to delimit the condition number is a way to overcome this. This prevents the ES from evolving in a degenerated subspace of the null space when approaching the boundary. The regularization is done by adding a small positive value to the diagonal elements if the condition number exceeds a threshold t, i.e.,

$$\mathbf{M}_r = \sqrt{\mathbf{C}} + r\mathbf{I} \text{ with } r = 0 \text{ if } \text{cond}(\mathbf{C}) \leq t. \tag{12}$$

The regularized covariance matrix $\tilde{\mathbf{C}}$ is then $\mathbf{M}_r\mathbf{M}_r^T$. Let λ_i denote the i-th eigenvalue[3] of $\mathbf{C}$ such that $\lambda_1 \leq \lambda_i \leq \lambda_N$. Accordingly, the i-th eigenvalue of $\sqrt{\mathbf{C}}$ is $\sqrt{\lambda_i}$. The eigenvalues of $\mathbf{M}_r$ and $\mathbf{M}_r\mathbf{M}_r^T$ are $\sqrt{\lambda_i}+r$ and $(\sqrt{\lambda_i}+r)^2$, respectively.

In case the condition number exceeds the threshold t, i.e., $\text{cond}(\mathbf{C}) = \text{cond}(\mathbf{M}\mathbf{M}^T) = \lambda_N/\lambda_1 > t$, the factor r is chosen to limit the condition number to t. That is, the corresponding r value is determined by

$$\text{cond}(\tilde{\mathbf{C}}) = \text{cond}(\mathbf{M}_r\mathbf{M}_r^T) = \frac{(\sqrt{\lambda_N}+r)^2}{(\sqrt{\lambda_1}+r)^2} \stackrel{!}{=} t \tag{13}$$

The detailed steps solving Eq. (13) for r are provided in the supplementary material (Sec. VI-A). They result in

$$r = \frac{\sqrt{\lambda_N}}{t} - \sqrt{\lambda_1} + \sqrt{\frac{\lambda_N}{t^2} + \frac{\lambda_N}{t} - \frac{2\sqrt{\lambda_1\lambda_N}}{t}}. \tag{14}$$

IV. Experimental Evaluation

The lcCMSA-ES is tested on a variety of different test functions. Linear objective functions are considered as a first step and non-linear objective functions as a second step. For the linear objective function tests, the Klee-Minty cube [37] is used. For the non-linear objective function experiments, the BBOB COCO framework [38] with adaptions is used. In addition, the performance of the lcCMSA-ES is compared with other methods that are able to deal with problem (1).

[3]Note that we use λ_i here to denote an eigenvalue. In Alg. 3 we use λ to denote the number of offspring.

Algorithm 3 The $(\mu/\mu_I, \lambda)$-lcCMSA-ES.

1: Input $\mathbf{A}, \mathbf{b}, f$
2: Initialize parameters μ, λ, σ, τ, τ_c, G, G_{lag}, g_{stop}, σ_{stop}, ϵ_{abs}, ϵ_{rel}, t
3: $\mathbf{B}^{D\times N} \leftarrow$ orthonormalize(null($\mathbf{A}$))
4: $\mathbf{C} \leftarrow \mathbf{I}^{N\times N}$
5: $\mathbf{x}^{(0)} \leftarrow$ findInhomogeneousSolution($\mathbf{A}, \mathbf{b}$)
6: $P \leftarrow$ initReferencePointsForIterativeProjection($\mathbf{x}^{(0)}$, $10N$, $\mathbf{A}$, $\mathbf{b}$)
7: Randomize $\mathbf{x}^{(0)}$,
e.g.: $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(0)} + ||\mathbf{x}^{(0)}||\mathbf{B}\mathcal{N}(\mathbf{0}, \mathbf{I}^{N\times N})$
8: **if** $\left(\mathbf{x}^{(0)}\right)_{1:D} < 0$ **then**
9: $\mathbf{x}^{(0)} \leftarrow$ projectToPositiveOrthantIter($\mathbf{x}^{(0)}$, $\mathbf{A}$, $\mathbf{b}$, P)
10: **end if**
11: $(\mathfrak{a}_{\text{bsf}}, g_{\text{bsf}}) \leftarrow \left((f(\mathbf{x}^{(0)}), \mathbf{x}^{(0)}, \mathbf{0}, \mathbf{0}, \sigma), 0\right)$
12: $g \leftarrow 0$
13: **repeat**
14: $(\sqrt{\mathbf{C}})_{\text{normalized}} \leftarrow$ computeSqrtCNormalized($\mathbf{C}$, t)
15: **for** $l \leftarrow 1$ **to** λ **do**
16: $\tilde{\sigma}_l \leftarrow \sigma e^{\tau\mathcal{N}_l(0,1)}$
17: $\tilde{\mathbf{s}}_l \leftarrow (\sqrt{\mathbf{C}})_{\text{normalized}}\mathcal{N}_l(\mathbf{0}, \mathbf{I}^{N\times N})$
18: $\tilde{\mathbf{z}}_l \leftarrow \tilde{\sigma}_l\mathbf{B}\tilde{\mathbf{s}}_l$
19: $\tilde{\mathbf{x}}_l \leftarrow \mathbf{x}^{(g)} + \tilde{\mathbf{z}}_l$
20: **if** $(\tilde{\mathbf{x}}_l)_{1:D} < 0$ **then**
21: $\tilde{\mathbf{x}}_l \leftarrow$ projectToPositiveOrthantIter($\tilde{\mathbf{x}}_l$, $\mathbf{A}$, $\mathbf{b}$, P)
22: $\tilde{\mathbf{z}}_l \leftarrow \tilde{\mathbf{x}}_l - \mathbf{x}^{(g)}$
23: $\tilde{\mathbf{s}}_l \leftarrow \mathbf{B}^T\tilde{\mathbf{z}}_l/\tilde{\sigma}_l$
24: **end if**
25: $\tilde{f}_l \leftarrow f(\tilde{\mathbf{x}}_l)$
26: $\tilde{\mathfrak{a}}_l \leftarrow (\tilde{f}_l, \tilde{\mathbf{x}}_l, \tilde{\mathbf{z}}_l, \tilde{\mathbf{s}}_l, \tilde{\sigma}_l)$
27: $(\mathfrak{a}_{\text{bsf}}, g_{\text{bsf}}) \leftarrow \begin{cases} (\tilde{\mathfrak{a}}_l, g+1) & \text{if } \tilde{\mathfrak{a}}_l \succ \mathfrak{a}_{\text{bsf}} \\ (\mathfrak{a}_{\text{bsf}}, g_{\text{bsf}}) & \text{otherwise} \end{cases}$
28: **end for**
29: rankOffspringPopulation($\tilde{\mathfrak{a}}_1, \ldots, \tilde{\mathfrak{a}}_\lambda$) acc. to "$\succ$"(Eq. (11))
30: $\mathbf{x}^{(g+1)} \leftarrow \mathbf{x}^{(g)} + \langle\tilde{\mathbf{z}}\rangle$
31: $\mathfrak{a} \leftarrow \left(f(\mathbf{x}^{(g+1)}), \mathbf{x}^{(g+1)}, \langle\tilde{\mathbf{z}}\rangle, \langle\tilde{\mathbf{s}}\rangle, \langle\tilde{\sigma}\rangle\right)$
32: $(\mathfrak{a}_{\text{bsf}}, g_{\text{bsf}}) \leftarrow \begin{cases} (\mathfrak{a}, g+1) & \text{if } \mathfrak{a} \succ \mathfrak{a}_{\text{bsf}} \\ (\mathfrak{a}_{\text{bsf}}, g_{\text{bsf}}) & \text{otherwise} \end{cases}$
33: $\sigma \leftarrow \langle\tilde{\sigma}\rangle$
34: $\mathbf{C} \leftarrow \left(1 - \frac{1}{\tau_c}\right)\mathbf{C} + \frac{1}{\tau_c}\langle\tilde{\mathbf{s}}\tilde{\mathbf{s}}^T\rangle$
35: $g \leftarrow g + 1$
36: **until** $g > g_{\text{stop}} \vee \sigma < \sigma_{\text{stop}} \vee ||\mathbf{x}^{(g)} - \mathbf{x}^{(g-G)}|| < \epsilon_{\text{abs}} \vee \left|\frac{||\mathbf{x}^{(g)}||}{||\mathbf{x}^{(g-G)}||} - 1\right| < \epsilon_{\text{rel}} \vee g - g_{\text{bsf}} \geq G_{\text{lag}}$

The algorithms are implemented in Octave with mex-extensions[4] and the experiments were run on a cluster with 5 nodes. Every node has an Intel 8-core Xeon E5420 2.50GHz processor with 8GiB of RAM running a GNU/Linux system. For the BBOB COCO tests, the post-processing tool with slight adjustments was used to generate the figures. This post-processing tool is part of the BBOB COCO framework.

[4]We provide the code in a GitHub repository (https://github.com/patsp/lcCMSA-ES).

Algorithm 4 Computation of $(\sqrt{\mathbf{C}})_{\text{normalized}}$.

function computeSqrtCNormalized($\mathbf{C}$, t)
  $\mathbf{C} \leftarrow \frac{1}{2}\left(\mathbf{C} + \mathbf{C}^T\right)$
  Perform eigendecomposition to get $\mathbf{U}$ and $\mathbf{D}$ such that $\mathbf{C} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ with $\mathbf{D}$ being the diagonal matrix of eigenvalues and the columns of $\mathbf{U}$ being the corresponding eigenvectors
  $(\lambda_1, \ldots, \lambda_N)^T \leftarrow \text{diag}(\mathbf{D})$
  $r \leftarrow 0$
  if $\text{cond}(\mathbf{C}) > t$ **then**
    $r = \frac{\sqrt{\lambda_N}}{t} - \sqrt{\lambda_1} + \sqrt{\frac{\lambda_N}{t} + \frac{\lambda_N}{t^2} - \frac{2\sqrt{\lambda_1\lambda_N}}{t}}$
  end if
  $\mathbf{M}_r \leftarrow \mathbf{U}\sqrt{\mathbf{D}} + r\mathbf{I}$
  $(\sqrt{\mathbf{C}})_{\text{normalized}} = \det(\mathbf{M}_r)^{-\frac{1}{N}}\mathbf{M}_r$
  return$((\sqrt{\mathbf{C}})_{\text{normalized}})$
end function

TABLE I
PARAMETER SETTINGS FOR THE LCCMSA-ES EXPERIMENTS.

Core ES param.		Stopping criteria param.	
λ	$4D$	G	10
μ	$\lfloor \frac{\lambda}{4} \rfloor$	G_{lag}	$50N$
σ (initial value)	$\frac{1}{\sqrt{D}}$	g_{stop}	10000
τ	$\frac{1}{\sqrt{2N}}$	σ_{stop}	10^{-6}
τ_c	$1 + \frac{N(N-1)}{2\mu}$	ϵ_{abs}	10^{-9}
t	10^{12}	ϵ_{rel}	10^{-9}

In the experiments, the parameters for the lcCMSA-ES are set as shown in Table I. The six parameters G, G_{lag}, g_{stop}, σ_{stop} ϵ_{abs}, and ϵ_{rel} are used for the stopping criteria. The chosen values turned out to be good choices in initial experiments. The initial σ, τ, and τ_c were set according to the suggestions in [1]. The parameters μ and λ were chosen to have a truncation ratio $\mu/\lambda = 1/4$ (similar as in [1]). The value of t was set as a trade-off between numerical accuracy and the toleration of approaching the boundary in the ES.

The sum of objective and constraint function evaluations are considered for the performance measure. In the BBOB COCO framework one call to the constraint evaluation function yields the values of all the constraints for a given query point.

A. Performance on the Klee-Minty cube

Klee and Minty formulated a special LP [37] to show that the Simplex algorithm [39], although working well in practice, has an exponential runtime in the worst case. To this end, they invented the so-called Klee-Minty cube. The n-dimensional Klee-Minty cube is a distorted hypercube with 2^n corners. The inside of the cube represents the feasible region. The objective function is constructed in such a way that the Simplex algorithm visits all the corners in the worst case and thus its worst case runtime is exponential. Formally, the inequalities of the feasible region write

$$\begin{array}{rcl} x_1 & & \leq 5 \\ 4x_1 + & x_2 & \leq 25 \\ \vdots & \vdots & \vdots \ \vdots \\ 2^n x_1 + 2^{(n-1)} x_2 + \cdots + 4x_{n-1} + x_n & & \leq 5^n \end{array} \tag{15}$$

where $x_1 \geq 0, \ldots, x_n \geq 0$. The objective function is $2^{(n-1)}x_1 + 2^{(n-2)}x_2 + \cdots + 2x_{n-1} + x_n \rightarrow \text{max!}$. The maximum is reached for the vector $\mathbf{x}_{\text{opt}} = (0, 0, \ldots, 0, 5^n)^T$ yielding $f(\mathbf{x}_{\text{opt}}) = 5^n$. The Klee-Minty problem has been chosen because it is known to be also a hard problem for interior point methods [40], [41].

Table II shows the results of single runs of the lcCMSA-ES with the Iterative Projection on the Klee-Minty problem with different dimensions. The optimal value is reached up to a small error for all the dimensions from 1 to 15. For dimensions larger than 15 we have observed numerical instabilities.

We also tested interior point LP solvers on the Klee-Minty problem. We applied glpk's interior point LP algorithm using Octave and Mathematica's interior point LP algorithm to the Klee-Minty problem. We have observed that the absolute error to the optimum increases with increasing dimension for both LP solvers. The supplementary material contains detailed results (Sec. VI-E3). Tables III and IV display the results of single runs of the glpk LP solver and the Mathematica LP solver, respectively. Both solvers were run with default parameters and interior point methods. The number of generations and the number of function evaluations are not comparable. For example, according to the documentation, the default number of maximum iterations for the interior point algorithm glpk in Octave is 200. This is independent of the number of variables and constraints.

B. Performance on the BBOB COCO constrained suite

For the non-linear objective function experiments the BBOB COCO framework [38] with adaptions is used. The adapted version[5] is based on the code in the branch `development`[6] in [42]. A documentation can be found in [43] under `docs/bbob-constrained/functions/build` after building it according to the instructions.

The BBOB COCO framework provides a test suite, bbob-constrained, for constrained black-box optimization benchmarking. It contains 48 constrained functions with dimensions $D \in \{2, 3, 5, 10, 20, 40\}$. For every problem, random instances can be generated. The 48 problems are constructed by combining 8 functions of the standard BBOB COCO suite for single-objective optimization with 6 different numbers of constraints, namely 1, 2, 6, $6 + D/2$, $6 + D$, and $6 + 3D$

[5]We provide the adapted code in a GitHub fork of the BBOB COCO framework, https://github.com/patsp/coco. The changes are in the new branch `development-sppa-2`. This branch is based on the `development` branch of https://github.com/numbbo/coco with changes up to and including Dec 10, 2017. A list of the changes is also provided in the supplementary material (Sec. VI-F).

[6]Because the `bbob-constrained` suite is still under development, we provide a fork. This makes our results reproducible. Even though it is still under development, this suite gives a good indication of the algorithm's performance in comparison to other methods. We use this suite instead of defining our own test problems with linear constraints for this work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2018.2871944, IEEE Transactions on Evolutionary Computation

TABLE II
RESULTS OF SINGLE RUNS OF THE LCCMSA-ES WITH THE ITERATIVE PROJECTION (LINEAR RUNTIME VERSION) ON THE KLEE-MINTY CUBE.

Name	f_{opt}	$\text{ES}_{f_{\text{best}}}$	$\lvert f_{\text{opt}} - \text{ES}_{f_{\text{best}}}\rvert$	$\lvert f_{\text{opt}} - \text{ES}_{f_{\text{best}}}\rvert / \lvert f_{\text{opt}}\rvert$	#generations	#f-evals
Klee-Minty $D = 1$	-5.000000	-5.000000	2.910383e-11	5.820766e-12	97	874
Klee-Minty $D = 2$	-25.000000	-25.000000	2.693810e-10	1.077524e-11	104	1769
Klee-Minty $D = 3$	-125.000000	-125.000000	1.987161e-09	1.589729e-11	153	3826
Klee-Minty $D = 4$	-625.000000	-625.000000	2.280285e-08	3.648456e-11	201	6634
Klee-Minty $D = 5$	-3125.000000	-3125.000000	2.121087e-07	6.787479e-11	251	10292
Klee-Minty $D = 6$	-15625.000000	-15625.000003	2.568122e-06	1.643598e-10	301	14750
Klee-Minty $D = 7$	-78125.000000	-78125.000030	3.049150e-05	3.902912e-10	351	20008
Klee-Minty $D = 8$	-390625.000000	-390625.000303	3.030710e-04	7.758617e-10	403	26196
Klee-Minty $D = 9$	-1953125.000000	-1953125.001656	1.656145e-03	8.479462e-10	451	32924
Klee-Minty $D = 10$	-9765625.000000	-9765625.000914	9.139776e-04	9.359131e-11	501	40582
Klee-Minty $D = 11$	-48828125.000000	-48828125.000000	0.000000e+00	0.000000e+00	551	49040
Klee-Minty $D = 12$	-244140625.000000	-244140625.000000	2.980232e-08	1.220703e-16	602	58395
Klee-Minty $D = 13$	-1220703125.000000	-1220703125.000000	0.000000e+00	0.000000e+00	650	68251
Klee-Minty $D = 14$	-6103515625.000000	-6103515625.000001	9.536743e-07	1.562500e-16	735	83056
Klee-Minty $D = 15$	-30517578125.000000	-30517578125.000004	3.814697e-06	1.250000e-16	755	91356

constraints. The 8 functions are Sphere, Separable Ellipsoid, Linear Slope, Rotated Ellipsoid, Discus, Bent Cigar, Sum of Different Powers, and the Separable Rastrigin. The constraints are linear with nonlinear perturbations and defined by their gradient. These constraints are generated by sampling their gradient vectors from a normal distribution and ensuring that the feasible region is not empty. The generic algorithm of generating a constrained problem is outlined in [43], `docs/bbob-constrained/functions/build`.

The optimization problem in the BBOB COCO framework is stated as

$$f'(\mathbf{x}) \to \min! \tag{16a}$$
$$\text{s.t. } \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{16b}$$
$$\check{\mathbf{x}} \leq \mathbf{x} \leq \hat{\mathbf{x}} \tag{16c}$$

where $f' : \mathbb{R}^{D'} \to \mathbb{R}$ and $\mathbf{g} : \mathbb{R}^{D'} \to \mathbb{R}^{K'}$. In order for the lcCMSA-ES to be applicable this must be transformed into

$$f(\mathbf{x}) \to \min! \tag{17a}$$
$$\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \tag{17b}$$
$$\mathbf{x} \geq \mathbf{0} \tag{17c}$$

where $f : \mathbb{R}^D \to \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{K \times D}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^K$. It is known that the constraints in the bbob-constrained suite of the BBOB COCO framework are linear with non-linear perturbations. Using this fact in addition with the enhanced ability to disable the non-linear perturbations, a pre-processing step is used. It transforms Eq. (16) into Eq. (17). This pre-processing step is based on the idea of querying the constraint function at enough positions in the parameter space. This allows constructing a system of equations that can be solved for the underlying coefficients of the linear constraints. The resulting coefficients and the bounds can be put into matrix form. Slack variables are added for transforming the inequalities into equalities to arrive at the form in Eq. (17). Due to space limitations, Secs. VI-C and VI-D in the supplementary material describe how this can be done and show pseudo-code.

In the following, the performance of different algorithms is visualized by use of bootstrapped Empirical Cumulative Distribution Functions (ECDF). These plots show the percentages of function target values reached for a given budget of function and constraint evaluations per search space dimensionality. The x-axis shows the sum of objective function and constraint evaluations normalized by dimension (log-scaled). The y-axis shows the percentage of so-called targets that were reached for the given sum of objective function and constraint evaluations. Every target is defined as a particular distance from the optimum. In the plots shown, the standard BBOB ones are used: $f_{\text{target}} = f_{\text{opt}} + 10^k$ for 51 different values of k between -8 and 2. The crosses indicate the medians of the sum of objective function and constraint evaluations of instances that did not reach the most difficult target. Note that the steps at the beginning of the lines of some variants are due to the pre-processing step that requires an initial amount of constraint evaluations. Furthermore, the top-left corner in every plot shows information about the experiments. The first line indicates the functions of the BBOB COCO framework that have been used in the experiment. The second line specifies the targets. The number of runs (instances) are indicated in the third line. A line (with a marker) to every entry in the legend is drawn. This shows which line in the plot belongs to which entry in the legend.

The ECDF plots shown in Figs. 1 and 2 show results aggregated over multiple problems. For this, the results of an algorithm over all the problems are considered for a specific dimension. It is referred to [44] for all the details. In the supplementary material, we provide single function plots (Figs. 5 and 6).

Fig. 1 presents the ECDF of runs of the lcCMSA-ES with the Iterative Projection. All the constrained problems of the BBOB COCO bbob-constrained test suite are shown aggregated. The considered dimensions are 2, 3, 5, 10, 20, 40. For these, the performance of the algorithm is evaluated on 15 independent randomly generated instances of each constrained test problem. Based on the observed run lengths, ECDF graphs are generated. Every line corresponds to the aggregated ECDF over all problems of a dimension (2, 3, 5, 10, 20, 40 from top to bottom).

Additional simulation results of the lcCMSA-ES are presented in the supplementary material (Sec. VI-E2). They include ECDF and average runtime graphs of the lcCMSA-ES for all the single functions of the BBOB COCO constrained

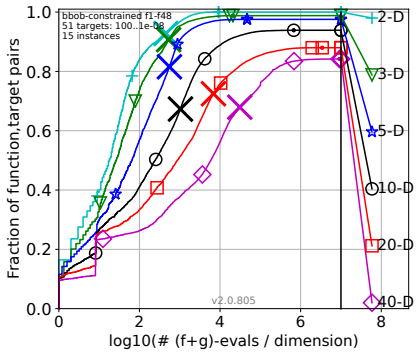

Fig. 1. Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension for the lcCMSA-ES with the Iterative Projection.

suite. Graphs showing the evolution dynamics of the lcCMSA-ES are presented as well.

We see that for the dimensions 2, 3, and 5 the most difficult target is reached with about $10^5 D$ function and constraint evaluations. For the higher dimensions the most difficult target is not reached. But about 90% of the targets are reached with about $10^6 D$ function and constraint evaluations. One can see that the performance in the higher dimensions is low in particular for the Rastrigin functions (functions 43-48 shown in the last six subplots of Fig. 6). The Rastrigin function is multimodal. In order to deal with such a function, an extension of the lcCMSA-ES is possible. An example could be an integration of the lcCMSA-ES into a restart meta ES.

C. Comparison with other approaches

To compare the lcCMSA-ES proposed in this work a selection of other algorithms is benchmarked on the same adapted BBOB COCO suite. Three variants of DE that showed promising results in benchmarks are tested, namely "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization" (conSaDE) [20], "Differential Evolution with Ensemble of Constraint Handling Techniques" (ECHT-DE) [21], and "Constrained Optimization by the ε Constrained Differential Evolution with an Archive and Gradient-Based Mutation" (εDEag) [22]. Further, an Active-Set ES [33], an ES with augmented Lagrangian constraint handling [34] and a method based on surrogate modeling with adaptive parameter control [45] (SACOBRA) are benchmarked and compared to the approach presented in this work.

For the conSaDE, the ECHT-DE, the εDEag, the Active-Set ES, the ES with augmented Lagrangian constraint handling, and the SACOBRA algorithm, the implementations provided by the respective authors were used[7] (adapted for the BBOB COCO framework). All the algorithms for the comparison were run with default parameters.

[7]conSaDE: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Codes/2006-CEC-Const-SaDE.rar, ECHT-DE: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Codes/2010-TEC-Ens-Con-EP-DE.zip, εDEag: http://www.ints.info.hiroshima-cu.ac.jp/%7etakahama/download/eDEa-2010.0430.tar.gz, Active-Set ES: https://web.cs.dal.ca/%7edirk/AS-ES.tar, SACOBRA: https://cran.r-project.org/web/packages/SACOBRA/index.html, ES with augmented Lagrangian constraint handling: Code provided by Asma Atamna.

The goal is to have a direct comparison to the method proposed in this work. The non-linear transformations are turned off in order to be able to compare the approaches to the lcCMSA-ES. For the lcCMSA-ES the inequality constraints are first pre-processed to transform the problem into one that the lcCMSA-ES is able to handle. Hence, the lcCMSA-ES solves (17). Similarly, for the active-set ES, the linear constraints are determined but no slack variables are added because inequalities can be handled by the algorithm. The optimization problem

$$f(\mathbf{x}) \rightarrow \min! \tag{18a}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{18b}$$

$$\check{\mathbf{x}} \leq \mathbf{x} \leq \hat{\mathbf{x}} \tag{18c}$$

is passed to the active-set ES, i.e., the active-set ES solves (18). $\mathbf{A}$ and $\mathbf{b}$ represent the BBOB COCO constraint system of the current problem and f is the current problem's objective function. As the DE variants, the CMA with augmented Lagrangian handling and the SACOBRA are able to handle the form of the BBOB COCO problem directly, no problem transformation is necessary for them. Therefore, they solve (16).

Fig. 2 shows ECDF graphs for all the different algorithms. For every algorithm the ECDF aggregated over all functions and dimensions in the BBOB COCO constrained suite is displayed. Our approach (named `itprojlccmsaes` for the variant with Iterative Projection and `l1lpsolvelccmsaes` for the variant with the ℓ_1 projection in the plots) is among the best for all the dimensions. The CMA-ES with augmented Lagrangian constraint handling (named `cma_es_augmented_lagrangian` in the plots) is able to reach about 50-60% of the targets. The Active-Set ES (named `activesetES` in the plots) performs similarly to our approach for all the dimensions. The DE variants, the Active-Set ES, and the SACOBRA approach perform similarly as our approach in the smaller dimensions but not as well in the larger dimensions. The exception to this is the conSaDE that performs better for dimension 20 and similarly as our algorithm for dimension 40. Similar to Fig. 1, a closer look at the single function ECDF plots for all the algorithms (not shown here) reveals more insight. The lower performance in the higher dimensions 20 and 40 is mainly due to the Rastrigin problem for all the algorithms. The CMA-ES with augmented Lagrangian constraint handling is only able to solve a subset of the problems for all dimensions. In particular, it is able to solve the Sphere, the Linear Slope, and the Different Powers problems with 1, 2, and 6 constraints. The SACOBRA algorithm has problems with the higher number of constraints as well.

Fig. 3 shows ECDF graphs for all the different algorithms for the Klee-Minty problem with dimensions 9, 12, and 15 (plots for all the dimensions are presented in the supplementary material (Sec. VI-E3)). The SACOBRA approach, the εDEag, and the CMA with augmented Lagrangian constraint handling are only able to reach 20% of the targets. All the other approaches perform well with some difficulties in higher dimensions (13-15). In the large dimensions it is worth noting that the numbers get large quickly and therefore numerical stability can become an issue.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2018.2871944, IEEE Transactions on Evolutionary Computation

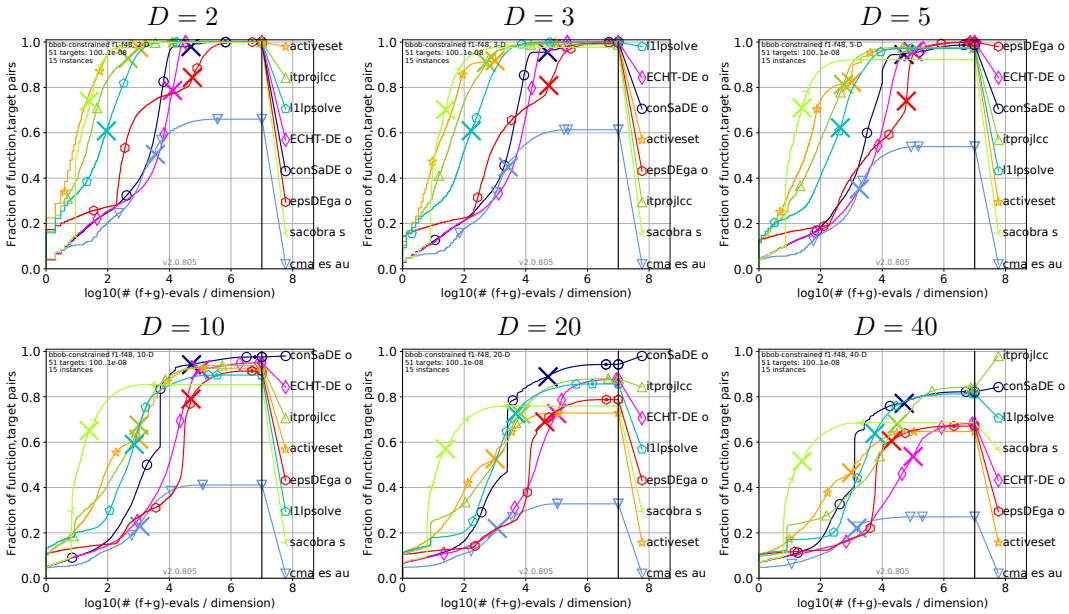

Fig. 2. Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of all the approaches.

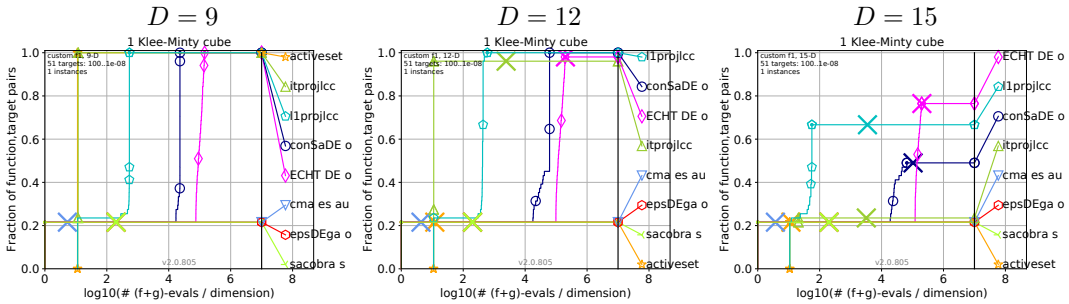

Fig. 3. Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension for the Klee-Minty problem with dimensions $D = 9$, $D = 12$ and $D = 15$: comparison of all the approaches.

V. Conclusion

We have proposed the lcCMSA-ES; a CMSA-ES for solving optimization problems with linear constraints. The algorithm is based on the CMSA-ES. It is an interior point approach that repairs infeasible candidate solutions if necessary. The mutation operator and repair method are specially designed. They allow the ES to evolve itself on a linear manifold. This distinguishes the proposed algorithm from the other methods considered for the comparison. It has been experimentally shown that the method works well on the Klee-Minty optimization problem as well as the bbob-constrained suite of the BBOB COCO framework (with disabled non-linear perturbations). Additionally, the proposed lcCMSA-ES has been compared to other evolutionary approaches for constrained optimization. Experiments have shown that the lcCMSA-ES is among the best for the BBOB COCO constrained suite and the Klee-Minty problem. It is worth noting that not all algorithms that have been compared are interior point methods. Consequently, they have the advantage of evaluating the objective function outside the feasible region. In particular, they do not move on the linear manifold defined by the constraints. All the three DE variants considered (conSaDE, ECHT-DE, and εDEag) allow infeasible candidates. The ES with augmented Lagrangian constrained handling works with a penalty. Thus, infeasible candidates are involved during the evolution as well. The surrogate modeling with adaptive parameter control does most of the optimization work on the surrogate models. But already the computation of the initial surrogate models involves evaluating objective and constraint functions. For the initialization this is done at random points in the search space. Those random points are not necessarily feasible. The optimization on the surrogate models involves infeasible solutions with respect to the real functions. But the results of the optimization on the surrogate models are tried to be repaired if necessary with respect to the real functions. The Active-Set ES only considers feasible candidates. It starts with an initial feasible candidate solution and uses repair by projection for dealing with infeasible solutions. This repair approach is not designed to move on the linear space defined by the constraints. This is in contrast to our proposed ES. Consequently, the objective function is only evaluated for feasible candidate solutions.

REFERENCES

[1] H.-G. Beyer and B. Sendhoff, "Covariance matrix adaptation revisited – the CMSA evolution strategy –," in *Parallel Problem Solving from Nature – PPSN X: 10th International Conference, Dortmund, Germany, September 13-17, 2008. Proceedings*. Berlin, Heidelberg: Springer, 2008, pp. 123–132.

[2] H.-G. Beyer and S. Finck, "On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 578–596, Aug 2012.

[3] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, Jun. 2001.

[4] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[5] B. A. McCarl and T. H. Spreen, *Applied Mathematical Programming using algebraic systems*. Cambridge, MA, 1997.

[6] G. C. Pflug and W. Römisch, *Modeling, measuring and managing risk*. World Scientific, 2007.

[7] W. Yan, R. Miao, and S. Li, "Multi-period semi-variance portfolio selection: Model and numerical solution," *Applied Mathematics and Computation*, vol. 194, no. 1, pp. 128–134, 2007.

[8] M. S. Kaylen, P. V. Preckel, and E. T. Loehman, "Risk modeling via direct utility maximization using numerical quadrature," *American Journal of Agricultural Economics*, vol. 69, no. 3, pp. 701–706, 1987.

[9] B. Baumrucker and L. Biegler, "MINLP & MPCC strategies for optimization of a class of hybrid dynamic systems," 2009.

[10] J. Buijs, J. Ludlage, W. V. Brempt, and B. D. Moor, "Quadratic programming in model predictive control for large scale systems," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 301 – 306, 2002, 15th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667015387218

[11] J. Castro, "Minimum-distance controlled perturbation methods for large-scale tabular data protection," *European Journal of Operational Research*, vol. 171, no. 1, pp. 39 – 52, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221704006113

[12] M. Gorji-Bandpy, H. Yahyazadeh-Jelodar, and M. Khalili, "Optimization of heat exchanger network," *Applied Thermal Engineering*, vol. 31, no. 5, pp. 779–784, 2011.

[13] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Tech. Rep., 2011. [Online]. Available: http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC11-RWP/CEC11-RWP.htm

[14] J. Bracken and G. P. McCormick, *Selected Applications of Nonlinear Programming*. John Wiley & Sons, Inc., 1968.

[15] M. H. Wright, "Direct search methods: Once scorned, now respectable," in *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, ser. Pitman Research Notes in Mathematics, D. F. Griffiths and G. A. Watson, Eds., vol. 344. CRC Press, 1996, pp. 191–208.

[16] M. W. Trosset, "I know it when I see it: Toward a definition of direct search methods," in *SIAG/OPT Views-and-News: A Forum for the SIAM Activity Group on Optimization*, vol. 9, 1997, pp. 7–10.

[17] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM review*, vol. 45, no. 3, pp. 385–482, 2003.

[18] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. SIAM, 2009.

[19] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[20] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2006, pp. 17–24.

[21] R. Mallipeddi and P. N. Suganthan, "Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–8.

[22] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–9.

[23] F. Neumann and I. Wegener, "Minimum spanning trees made easier via multi-objective optimization," *Natural Computing*, vol. 5, no. 3, pp. 305–319, Sep 2006.

[24] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," *Evolutionary Computation*, vol. 18, no. 4, pp. 617–633, 2010.

[25] C. Qian, Y. Yu, and Z.-H. Zhou, "On constrained boolean pareto optimization," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 389–395.

[26] I. Rechenberg, *Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973.

[27] H.-P. Schwefel, *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., 1981.

[28] H.-G. Beyer, *Ein Evolutionsverfahren zur mathematischen Modellierung stationärer Zustände in dynamischen Systemen*. Hochschule für Architektur und Bauwesen, 1989.

[29] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, Sep. 2000. [Online]. Available: http://dx.doi.org/10.1109/4235.873238

[30] E. Mezura-Montes and C. A. C. Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, Feb 2005.

[31] D. V. Arnold and N. Hansen, "A (1 + 1)-CMA-ES for constrained optimisation," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2012, pp. 297–304.

[32] R. Chocat, L. Brevault, M. Balesdent, and S. Defoort, "Modified covariance matrix adaptation – evolution strategy algorithm for constrained optimization under uncertainty, application to rocket design," *International Journal for Simulation and Multidisciplinary Design Optimization*, vol. 6, p. A1, 2015.

[33] D. V. Arnold, "An active-set evolution strategy for optimization with known constraints," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 192–202.

[34] A. Atamna, A. Auger, and N. Hansen, "Linearly convergent evolution strategies via augmented lagrangian constraint handling," in *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. ACM, 2017, pp. 149–161.

[35] S. Le Digabel and S. Wild, "A taxonomy of constraints in simulation-based optimization," Les cahiers du GERAD, Tech. Rep. G-2015-57, 2015.

[36] A. Kannan and S. M. Wild, "Benefits of deeper analysis in simulation-based groundwater optimization problems," in *Computational Methods in Water Resources (CMW 2012)*, University of Illinois at Urbana-Champaign, 06/2012 2012.

[37] V. Klee and G. J. Minty, "How good is the simplex algorithm?" in *Inequalities*, O. Shisha, Ed. New York: Academic Press, 1972, vol. III, pp. 159–175.

[38] S. Finck, N. Hansen, R. Ros, and A. Auger, *COCO Documentation, Release 15.03*. [Online]. Available: http://coco.gforge.inria.fr/

[39] G. B. Dantzig, "Maximization of a linear function of variables subject to linear inequalities," in *Activity Analysis of Production and Allocation*. New York: Wiley, 1951, ch. XXI.

[40] N. Megiddo and M. Shub, "Boundary behavior of interior point algorithms in linear programming," *Mathematics of Operations Research*, vol. 14, no. 1, pp. 97–146, 1989.

[41] A. Deza, E. Nematollahi, and T. Terlaky, "How good are interior point methods? Klee–Minty cubes tighten iteration-complexity bounds," *Mathematical Programming*, vol. 113, no. 1, pp. 1–14, 2008.

[42] D. Brockhoff, N. Hansen, O. Mersmann, R. Ros, D. Tusar, and T. Tusar, *COCO Code Repository*. [Online]. Available: http://github.com/numbbo/coco

[43] ——, *COCO Documentation Repository*. [Online]. Available: http://github.com/numbbo/coco-doc

[44] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, and T. Tusar, "COCO: Performance Assessment," May 2016, arXiv e-prints, arXiv:1605.03560. [Online]. Available: https://hal.inria.fr/hal-01315318

[45] S. Bagheri, W. Konen, M. Emmerich, and T. Bäck, "Solving the G-problems in less than 500 iterations: Improved efficient constrained optimization by surrogate modeling and adaptive parameter control," *CoRR*, 2015.