A Matrix Adaptation Evolution Strategy for Optimization on General Quadratic Manifolds

Patrick Spettel
Vorarlberg University of Applied Sciences
Research Center Business Informatics
Dornbirn, Austria
patrick.spettel@aon.at

Hans-Georg Beyer
Vorarlberg University of Applied Sciences
Research Center Business Informatics
Dornbirn, Austria
hans-georg.beyer@fhv.at

ABSTRACT

An evolution strategy design is presented that allows for an evolution on general quadratic manifolds. That is, it covers elliptic, parabolic, and hyperbolic equality constraints. The peculiarity of the presented algorithm design is that it is an interior point method. It evaluates the objective function only for feasible search parameter vectors and it evolves itself on the nonlinear constraint manifold. This is achieved by a closed form transformation of an individual's parameter vector, which is in contrast to iterative repair mechanisms. Results of different experiments are presented. A test problem consisting of a spherical objective function and a single hyperbolic/parabolic equality constraint is used. It is designed to be scalable in the dimension and it is used to compare the performance of the developed algorithm with other optimization methods supporting constraints. The experiments show the effectiveness of the proposed algorithm on the considered problems.

CCS CONCEPTS

• **Theory of computation** → **Bio-inspired optimization**; • **Computing methodologies** → **Randomized search**.

KEYWORDS

Matrix adaptation evolution strategy, nonlinear constraint, hyperbolic constraint, parabolic constraint, elliptic constraint, constraint handling by repair.

ACM Reference Format:
Patrick Spettel and Hans-Georg Beyer. 2021. A Matrix Adaptation Evolution Strategy for Optimization on General Quadratic Manifolds. In *Proceedings of The Genetic and Evolutionary Computation Conference 2021 (GECCO '21).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Constraint handling in evolutionary optimization methods is an important aspect in the area of black-box optimization. Many different constraint handling techniques have been proposed (for an overview, refer for example to [8]). There is a variety of differential evolution (DE) approaches for black-box optimization incorporating constraint handling [6, 7, 9, 12, 15]. An evolution strategy (ES) has been proposed [5] using similar constraint handling techniques as some of those DE methods. Whereas they allow evaluating the objective function outside of the feasible region, the focus of this work is the development of an interior point ES. As such, it represents a further step for a better understanding of constraint handling in ESs. Interior point methods are especially important in the case where it is not possible to evaluate infeasible parameter vectors. A prominent example in this regard is the area of simulation-based optimization (SBO), where it can be that there are inputs for which the simulator does not work[1] (e.g., limitations that stem from physics).

An interior point Constraint Matrix Self-Adaptation ES (CMSA-ES) supporting linear equality and inequality constraints has been proposed in [11]. The peculiarity of that method is that it evolves itself on the linear constraint manifold. This is achieved by a special constraint handling approach based on a repair method.

In [10], an interior point Matrix-Adaptation ES (MA-ES) with support for an elliptic equality constraint has been developed. It has been benchmarked on the Thomson problem [14] and one motivating problem is a real-world application in the area of financial stress-testing [3]. In that problem the consideration of only plausible scenarios creates an elliptic constraint.

The contribution of this work is the extension of the MA-ES to support elliptic equality constraints as well as parabolic or hyperbolic equality constraints. This extension, however, is not trivial: An elliptic constraint can be formulated by a positive definite quadratic form. This allows the application of the Cholesky decomposition, which can be used to develop the method described in [10] for ensuring that individuals are always feasible. A different design is required to deal with a parabolic or hyperbolic constraint. The goal is to develop an interior point method that ensures that individuals are feasible by a closed form transformation. The main motivation stems from the idea to support a quadratic form equality constraint with nonpositive eigenvalues.

The remainder of this work is organized as follows. Section 2 explains the form of the constraints that are supported in the algorithm designed in this work. Then, Section 3 describes the details about the algorithm design. After that, the experimental evaluation setup and the results are presented in Section 4. Finally, Section 5 provides a conclusion.

GECCO '21, July 10–14, 2021, Lille, France
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1]There are also problems for which the simulation must first be run in order to know whether a constraint is violated. Such problems are not the focus of this work.

2 PARABOLIC AND HYPERBOLIC CONSTRAINTS

The optimization problem considered in this work is

$$\text{min. } f(\mathbf{x}) \text{ s.t. } \mathbf{x}^T \mathbf{S}\mathbf{x} = \varkappa, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{S} \in \mathbb{R}^{N\times N}$, $\varkappa \in \mathbb{R}$, and no knowledge about f is assumed (i.e., f can be a black-box function). Additionally, in contrast to [10], there is no positive definiteness assumption of the constraint $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$. This has the direct consequence that S can have eigenvalues that are nonpositive. Hence, not only elliptic constraints, but also hyperbolic and parabolic constraints can be represented (revealing in eigenvalues that are negative or zero, respectively). Visualizations in two dimensions are provided in Appendix B.

3 ALGORITHM

3.1 Overview

The pseudo-code of the proposed algorithm is presented as Algorithm 1. It is based on the $(\mu/\mu_w, \lambda)$-MA-ES [2] and incorporates a closed-form equality constraint handling. Initialization and preprocessing steps are performed in lines 1 to 7. The details about the preprocessing are described in Section 3.3. Then, the generational loop is entered after initializing the generation counter (lines 8 to 9). In every generation, λ offspring are generated in lines 10 to 24 (the details of the offspring generation are explained in the following sections). Furthermore, notice that the evolution is performed in an $N + 1$ dimensional space. The reason is also explained in the following sections. The created offspring are then sorted in ascending order according to their fitness (because the problem is stated as a minimization problem) in line 25. Making use of the best μ offspring[2], the parental individual for the next generation is then computed (line 26). Notice that the offspring are ensured to be feasible by the closed form transformation that is derived in Section 3.2. In an implementation, one would also evaluate the objective function on the updated parental individual (line 26). For doing this, the same transformation as for a particular offspring has to be performed for the updated parental individual before the objective function evaluation. Finally, the update of the cumulation path $\mathbf{s}$, the covariance factor $\mathbf{M}$, the mutation strength σ, and the generation counter g ends one iteration of the generational loop. The matrix $\mathbf{M}$ and the mutation strength σ are updated according to [2].

3.2 Constraint Handling Idea

It turned out that the transformation used in [10] to transform the offspring to the elliptical manifold cannot be directly extended to general quadratic forms. To see the difference, we first recap the original transformation approach. Thereafter, the novel decomposition and projection idea will be presented.

3.2.1 Recap of the Method for Handling an Elliptic Equality Constraint. In [10], elliptical constraints have been assumed. That is, formally the constraint $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$ with $\mathbf{S} = \mathbf{S}^T$ and $\varkappa > 0$ has been considered. With this formulation, Cholesky decomposition can for example be used to get $\mathbf{S} = \mathbf{A}^T\mathbf{A}$. Now, an individual

$$\tilde{\mathbf{x}} = \mathbf{x} + \sigma\mathbf{z} \tag{2}$$

that potentially violates the equality constraint can be ensured to be feasible by modifying Equation (2) to

$$\tilde{\mathbf{x}} = \sqrt{\varkappa}\left(\frac{\mathbf{x} + \sigma\mathbf{A}^{-1}\mathbf{z}}{||\mathbf{A}\mathbf{x} + \sigma\mathbf{z}||}\right). \tag{3}$$

Note that $\mathbf{x}$ here denotes the parental individual, σ the mutation strength, and $\mathbf{z}$ the mutation vector. The update (3) ensures that the equality constraint is fulfilled for $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}}^T\mathbf{S}\tilde{\mathbf{x}} = \varkappa\frac{\left(\mathbf{x} + \sigma\mathbf{A}^{-1}\mathbf{z}\right)^T \mathbf{A}^T\mathbf{A}\left(\mathbf{x} + \sigma\mathbf{A}^{-1}\mathbf{z}\right)}{(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})^T\,(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})} \tag{4}$$

$$= \varkappa\frac{(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})^T\,(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})}{(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})^T\,(\mathbf{A}\mathbf{x} + \sigma\mathbf{z})} \tag{5}$$

$$= \varkappa. \tag{6}$$

3.2.2 How to ensure a Parabolic/Hyperbolic Equality Constraint. The approach with the decomposition described above works if S has only positive eigenvalues (elliptic constraint). For supporting S with nonpositive eigenvalues (parabolic/hyperbolic constraint), the main insight is to decompose the constraint

$$\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa =: -\varkappa_- + \varkappa_+. \tag{7}$$

Moreover, S is decomposed and together with projection operators, a closed form transformation for ensuring feasibility can be derived as follows. From the eigenvalue problem $\mathbf{S}\mathbf{u}_j = \lambda_j\mathbf{u}_j$ with $j \in \{1, \dots, N\}$ and $\mathbf{u}_j^T\mathbf{u}_k = \delta_{jk}$, one gets

$$\mathbf{S} = \sum_{j=1}^{N}\lambda_j\mathbf{u}_j\mathbf{u}_j^T = \underbrace{\sum_{i:\lambda_i<0}\lambda_i\mathbf{u}_i\mathbf{u}_i^T}_{=:\mathbf{S}_-} + \underbrace{\sum_{j:\lambda_j=0}\lambda_j\mathbf{u}_j\mathbf{u}_j^T}_{=:\mathbf{S}_0=\mathbf{0}} + \underbrace{\sum_{k:\lambda_k>0}\lambda_k\mathbf{u}_k\mathbf{u}_k^T}_{=:\mathbf{S}_+}. \tag{8}$$

In addition, one defines the projection operators

$$\mathbf{P}_- := \sum_{i:\lambda_i<0}\mathbf{u}_i\mathbf{u}_i^T, \mathbf{P}_0 := \sum_{j:\lambda_j=0}\mathbf{u}_j\mathbf{u}_j^T, \text{ and } \mathbf{P}_+ := \sum_{k:\lambda_k>0}\mathbf{u}_k\mathbf{u}_k^T. \tag{9}$$

Using Equation (8) and the definitions in (9), one can derive

$$\tilde{\mathbf{x}} := \left[\frac{\sqrt{\varkappa_-}}{\sqrt{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}}\mathbf{P}_- + \mathbf{P}_0 + \frac{\sqrt{\varkappa_+}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{P}_+\right]\mathbf{x}, \tag{10}$$

which is a nonlinear transformation of $\mathbf{x}$ to $\tilde{\mathbf{x}}$ ensuring that $\tilde{\mathbf{x}}$ is feasible (a proof is given in Appendix A). Notice that with this approach, there is one more free variable: Either $\varkappa_-$ or $\varkappa_+$ must be chosen. Making this choice a hyperparameter is problematic since different choices define different constraint manifolds. The approach followed here is to incorporate this variable into the evolution. For the details, it is referred to Section 3.4.

Equation (10) simplifies for the case of an elliptic constraint (only positive eigenvalues) and the case of a parabolic constraint (zero and positive but no negative eigenvalues). For the latter case, the simplification results in $\tilde{\mathbf{x}} \leftarrow \left(\frac{\sqrt{\varkappa}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{P}_+ + \mathbf{P}_0\right)\mathbf{x}$. For the former

[2]Note that $m;\lambda$ is the order statistic notation and represents the m-th best (w.r.t. fitness) out of λ values.

case, one gets $\tilde{\mathbf{x}} \leftarrow \frac{\sqrt{\varkappa}}{\sqrt{\mathbf{x}^T \mathrm{S}\mathbf{x}}}\mathbf{x}$. Note, this transformation is different to (3) used in [10].

3.3 Algorithmic Preprocessing for the Constraint Handling

In order to deal with the constraints as described in Section 3.2.2, a preprocessing consisting of three steps is performed (lines 1 to 7 in Algorithm 1) before entering the generational loop.

The first step of the preprocessing is to ensure that the matrix S is symmetric. For this, S is written as the sum

$$\mathrm{S} = \underbrace{\frac{\mathrm{S}+\mathrm{S}^T}{2}}_{=:\mathrm{S}_1} + \underbrace{\frac{\mathrm{S}-\mathrm{S}^T}{2}}_{=:\mathrm{S}_2}. \tag{11}$$

By taking a close look at the elements of $\left(\mathrm{S}+\mathrm{S}^T\right)/2$ and $\left(\mathrm{S}-\mathrm{S}^T\right)/2$, one notes that S_1 is symmetric and S_2 is antisymmetric, i.e., $\mathrm{S}_1^T = \mathrm{S}_1$ and $\mathrm{S}_2^T = -\mathrm{S}_2$, respectively. With these observations,

$$\mathbf{x}^T\mathrm{S}_2\mathbf{x} = \left(\mathbf{x}^T\mathrm{S}_2\mathbf{x}\right)^T = \mathbf{x}^T\mathrm{S}_2^T\mathbf{x} = -\mathbf{x}^T\mathrm{S}_2\mathbf{x} \tag{12}$$

follows. The first step is justified because $\mathbf{x}^T\mathrm{S}_2\mathbf{x}$ is a scalar. The second step makes use of the properties of the transpose calculation. The last step uses the antisymmetry of S_2. Using the result of the derivation, one gets $\mathbf{x}^T\mathrm{S}_2\mathbf{x} = 0$. With this,

$$\mathbf{x}^T\mathrm{S}_1\mathbf{x} = \varkappa \implies \underbrace{\mathbf{x}^T\mathrm{S}\mathbf{x}}_{=\varkappa} + \mathbf{x}^T\mathrm{S}^T\mathbf{x} = 2\varkappa \implies \mathbf{x}^T\mathrm{S}^T\mathbf{x} = \varkappa \tag{13}$$

can be derived. Hence, the transformation is justified for ensuring S to be symmetric and still $\mathbf{x}^T\mathrm{S}\mathbf{x} = \varkappa$ does hold (notice that with those observations and derivations, $\mathrm{S}_{\mathrm{sym}}$ corresponds to Equation (11)).

The second step is to perform the eigendecomposition $\mathrm{S}_{\mathrm{sym}} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ (line 4). The eigenvalues are denoted as $\lambda_1, \lambda_2, \ldots, \lambda_N$.

Third, the S_- and S_+ matrices for the decomposition and $\mathbf{P}_-$, $\mathbf{P}_0$, and $\mathbf{P}_+$ as the projection operators are computed.

3.4 Constraint Handling Implementation

With the preprocessing steps described in Section 3.3, the constraint handling can be performed as depicted in lines 14 to 21 for an offspring.

A single offspring is created by the standard MA-ES method (lines 11 to 13). However, due to the additional choice that stems from the decomposition in Equation (7), the evolution is performed in an $N+1$ dimensional space. This allows handling the additional variable as part of the evolution. In the proposed algorithm, $\varkappa_-$ is evolved as the $N+1$st component of an individual. It is denoted as $\tilde{\varkappa}_{-l}$. With this, one gets $\varkappa_+ = \tilde{\varkappa}_{-l} + \varkappa$ directly from Equation (7).

For the actual constraint handling in lines 14 to 21, three cases are considered: The first case of only positive eigenvalues corresponds to an elliptic constraint. The second case with zero and positive eigenvalues corresponds to a parabolic constraint. And the third case corresponds to a hyperbolic constraint. Notice that the decomposition in Equation (7) is only relevant for the third case.

Algorithm 1 The $(\mu/\mu_w, \lambda)$-MA-ES for optimization on elliptical, parabolic, and hyperbolic manifolds in $\mathbb{R}^N$, i.e., the supported constraint writes $\mathbf{x}^T\mathrm{S}\mathbf{x} = \varkappa$ with $\varkappa \in \mathbb{R}$, $\varkappa \geq 0$, $\mathrm{S} \in \mathbb{R}^{N\times N}$, and $\mathrm{S} \neq \mathbf{0}$, where the last condition excludes the case of all eigenvalues of S being 0. It is assumed that the feasible region is not empty. There is no assumption about the definiteness of S.

```
1:  Initialize parameters $\mu, \lambda, \mu_{\mathrm{eff}}, c_s, c_1, c_w$, and weights $w_m$ for $1 \leq m \leq \mu$ (for dim. $N+1$)
2:  Initialize $\mathbf{x}$ (dim. $N+1$) and $\sigma$
3:  $\mathbf{M} \leftarrow \mathbf{I}^{(N+1)\times(N+1)}$, $\mathbf{s} \leftarrow \mathbf{0}^{(N+1)\times 1}$, $\mathrm{S}_{\mathrm{sym}} \leftarrow (\mathrm{S}+\mathrm{S}^T)/2$
4:  Perform eigendecomposition $\mathrm{S}_{\mathrm{sym}} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ and let $\lambda_1, \ldots, \lambda_N$
            denote the eigenvalues of $\mathrm{S}_{\mathrm{sym}}$ (diagonal entries
            of the matrix $\mathbf{D}$).
5:  $\mathrm{S}_- \leftarrow \mathbf{0} + \sum_{j:\lambda_j<0} \lambda_j \mathbf{u}_j\mathbf{u}_j^T$, $\mathbf{P}_- \leftarrow \mathbf{0} + \sum_{j:\lambda_j<0} \mathbf{u}_j\mathbf{u}_j^T$
6:  $\mathrm{S}_+ \leftarrow \mathbf{0} + \sum_{j:\lambda_j>0} \lambda_j \mathbf{u}_j\mathbf{u}_j^T$, $\mathbf{P}_+ \leftarrow \mathbf{0} + \sum_{j:\lambda_j>0} \mathbf{u}_j\mathbf{u}_j^T$
7:  $\mathbf{P}_0 \leftarrow \mathbf{0} + \sum_{j:\lambda_j=0} \mathbf{u}_j\mathbf{u}_j^T$
8:  $g \leftarrow 0$
9:  repeat
10:     for $l \leftarrow 1$ to $\lambda$ do
11:         $\tilde{\mathbf{z}}_l \leftarrow \mathcal{N}_l(\mathbf{0}, \mathbf{I})$
12:         $\tilde{\mathbf{d}}_l \leftarrow \mathbf{M}\tilde{\mathbf{z}}_l$
13:         $\tilde{\mathbf{y}} \leftarrow (\mathbf{x} + \sigma\tilde{\mathbf{d}}_l)_{1..N}$
14:         if $\forall j \in \{1, \ldots, N\} : \lambda_j > 0$ then
                ▷ only positive eigenvalues
15:             $\tilde{\mathbf{x}}_l \leftarrow \frac{\sqrt{\varkappa}}{\sqrt{\tilde{\mathbf{y}}^T\mathrm{S}\tilde{\mathbf{y}}}}\tilde{\mathbf{y}}$
16:         else if $\nexists j \in \{1, \ldots, N\} : \lambda_j < 0$ then
                ▷ zero and positive eigenvalues
17:             $\tilde{\mathbf{x}}_l \leftarrow \left(\frac{\sqrt{\varkappa}}{\sqrt{\tilde{\mathbf{y}}^T\mathrm{S}_+\tilde{\mathbf{y}}}}\mathbf{P}_+ + \mathbf{P}_0\right)\tilde{\mathbf{y}}$
18:         else
                ▷ negative, positive, and (possibly) zero eigenvalues
19:             $\tilde{\varkappa}_{-l} \leftarrow |(\mathbf{x} + \sigma\tilde{\mathbf{d}}_l)_{N+1}|$
20:             $\tilde{\mathbf{x}}_l \leftarrow \left[\frac{\sqrt{\tilde{\varkappa}_{-l}}}{\sqrt{-\tilde{\mathbf{y}}^T\mathrm{S}_-\tilde{\mathbf{y}}}}\mathbf{P}_- + \mathbf{P}_0 + \frac{\sqrt{\tilde{\varkappa}_{-l}+\varkappa}}{\sqrt{\tilde{\mathbf{y}}^T\mathrm{S}_+\tilde{\mathbf{y}}}}\mathbf{P}_+\right]\tilde{\mathbf{y}}$
21:         end if
22:         $\tilde{f}_l \leftarrow f(\tilde{\mathbf{x}}_l)$
23:         $\tilde{\mathbf{x}}_l \leftarrow (\tilde{\mathbf{x}}_{l1}, \tilde{\mathbf{x}}_{l2}, \ldots, \tilde{\mathbf{x}}_{lN}, (\mathbf{x} + \sigma\tilde{\mathbf{d}}_l)_{N+1})^T$
24:     end for
25:     Sort offspring according to fitness in ascending order
26:     $\mathbf{x} \leftarrow \mathbf{x} + \sigma\sum_{m=1}^{\mu} w_m\tilde{\mathbf{d}}_{m;\lambda}$
27:     $\mathbf{s} \leftarrow (1-c_s)\,\mathbf{s} + \sqrt{\mu_{\mathrm{eff}}c_s\,(2-c_s)}\sum_{m=1}^{\mu} w_m\tilde{\mathbf{z}}_{m;\lambda}$
28:     $\mathbf{M} \leftarrow \mathbf{M}\left[\mathbf{I} + \frac{c_1}{2}\left(\mathbf{s}\mathbf{s}^T - \mathbf{I}\right) + \frac{c_w}{2}\left(\left(\sum_{m=1}^{\mu} w_m\tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^T\right) - \mathbf{I}\right)\right]$
29:     $\sigma \leftarrow \sigma\exp\left[\frac{c_s}{2}\left(\frac{||\mathbf{s}||^2}{N+1} - 1\right)\right]$
30:     $g \leftarrow g + 1$
31: until termination criteria fulfilled
```

4 EXPERIMENTAL EVALUATION

In [10], the Thomson Problem has been used for the experimental evaluation of the designed ES with support for an elliptic equality constraint. In this work, an artificially designed benchmark problem is used for a qualitative evaluation including a comparison with other approaches. Section 4.1 explains the problem for the qualitative analysis, Section 4.2 provides an overview of the algorithms that are used for comparison, Section 4.3 describes the experimental setup, and Section 4.4 presents and discusses the results.

4.1 The Problem Under Consideration

The idea is to have a benchmark problem that is scalable with respect to the dimensionality. Let N be the dimension. The proposed problem is the minimization of $f(\mathbf{x}) = \sum_{i=1}^{N/2} ((\mathbf{x})_i - 1)^2 + \sum_{j=N/2+1}^{N} (\mathbf{x})_j^2$ subject to

$$\mathbf{x}^T \begin{pmatrix} \mathbf{I}^{N/2\times N/2} & \mathbf{X} \\ N\mathbf{X}^T & -\mathbf{I}^{N/2\times N/2} \end{pmatrix} \mathbf{x} = N/2, \tag{14}$$

where $\left(\mathbf{X}^{N/2\times N/2}\right)_{ij} \sim \mathcal{N}(0,1)$. The objective function is a shifted sphere function such that the optimal parameter is

$$\mathbf{x}^* = (\underbrace{1,1,\ldots,1}_{N/2 \text{ ones}}, \underbrace{0,0,\ldots,0}_{N/2 \text{ zeros}})^T$$

with optimal objective function value $f(\mathbf{x}^*) = 0$. The constraint can intuitively be interpreted as the combination of two spherical constraints with added randomness.

Results of the proposed approach on further (randomized) problems are provided in the supplementary material (Appendix D).

4.2 Algorithms for the Comparison

The designed algorithm has been compared to the algorithms described in the following paragraphs.

ConSaDE (denoted "`conSaDE o`" in the plots): ConSaDE [6] is the so-called Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. It is based on the variant called Self-adaptive Differential Evolution (SaDE), which it extends by constraint handling using lexicographic ordering. It uses Matlab's fmincon as a local searcher.

ECHT-DE (denoted "`ECHT-DE o`" in the plots): Differential Evolution with Ensemble of Constraint Handling Techniques (ECHT-DE) [7] combines different constraint handling techniques into a DE algorithm (superiority of feasibility, self-adaptive penalty, ϵ-level constraint handling, and stochastic ranking).

ϵDEga (denoted "`epsDEga o`" in the plots): Constrained Optimization by the ϵ Constrained Differential Evolution with an Archive and Gradient-Based Mutation (ϵDEga) [12] uses the ϵ-level constraint handling method and constructs an archive for diversity handling. In addition, a gradient-based mutation method is applied.

Active-Set ES (denoted "`Active-Se`" in the plots): The Active-Set ES [1] incorporates the idea of using an active set in the optimization process into an ES. It is a $(1+1)$-ES creating one feasible offspring in every generation. This is then repaired by projection, which considers the active set. The mutation strength is adapted using the $1/5$th rule.

ϵMAg-ES (denoted "`epsMAg-ES`" in the plots): The ϵMAg-ES [5] is based on the MA-ES and makes use of three constraint handling techniques (reflection for the bounds, ϵ-level constraint handling, gradient-based repair).

MA-ES for General Nonlinear Equality Constraints (denoted "`maes nonl`" in the plots): The MA-ES for General Nonlinear Equality Constraints (refer to [10, Sec. 3]) is based on the MA-ES and uses an iterative repair mechanism for handling the constraints.

LSHADE44 (denoted "`LSHADE44`" in the plots): The algortihm "L-SHADE with Competing Strategies Applied to Constrained Optimization" (LSHADE44) [9] was the winner of the CEC 2017 competition on real-valued constrained optimization. It is based on the Success History Based DE (L-SHADE) [13]. Four variants of mutation and crossover operators are used and for constraint handling the lexicographic ordering approach is incorporated.

iUDE (denoted "`iUDE on c`" in the plots): The improved Unified Differential Evolution (iUDE) is based on the UDE [15] algorithm. It was the winner of the CEC 2018 competition on constrained optimization in the continuous domain[3]. A dual population approach and three different trial vector generation methods are incorporated. For the constraint handling, lexicographic and ϵ-level constraint handling are used.

fmincon (denoted "`fmincon o`" in the plots): The optimization function fmincon (interior-point) of Matlab has been used. This provides results of a further interior point algorithm in addition to the designed algorithm and the Active-Set ES.

4.3 Experimental Setup

For conducting the experiments, the problem described in Section 4.1 has been implemented in the COCO framework [4]. The corresponding source code is available in a GitHub fork of the COCO framework, https://github.com/patsp/coco. The adaptations and extensions can be found in the branch `development-sppa-manifold`, which is based on the branch `development` of https://github.com/numbbo/coco. The code for the problem described in Section 4.1 is part of the test suite called `custom` (`code-experiments/src/suite_custom.c`, `code-experiments/src/coco_suite.c`).

To be in line with the `bbob-constrained` COCO suite that only uses inequality constraints, the equality constraint $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$ is represented as two inequality constraints using an error tolerance of 10^{-8}. I.e., the two inequalities

$$-\left(\mathbf{x}^T\mathbf{S}\mathbf{x} - \varkappa\right) - 10^{-8} \le 0 \tag{15}$$

and

$$\left(\mathbf{x}^T\mathbf{S}\mathbf{x} - \varkappa\right) - 10^{-8} \le 0 \tag{16}$$

are used.

The default parameters stated in [2] have been used, i.e.:

$$\lambda = 4 + \lfloor 3 \ln N \rfloor,$$

$$\mu = \left\lfloor \frac{\lambda}{2} \right\rfloor,$$

$$w_m = \frac{\ln\left(\frac{\lambda+1}{2}\right) - \ln m}{\sum_{k=1}^{\mu}\left(\ln\left(\frac{\lambda+1}{2}\right) - \ln k\right)} \text{ for } 1 \le m \le \mu,$$

$$\mu_{\text{eff}} = \frac{1}{\sum_{m=1}^{\mu} w_m^2},$$

[3]The Matlab source code and a report of iUDE are available in the CEC 2018 competition materials.

$$c_s = \frac{\mu_{\text{eff}} + 2}{\mu_{\text{eff}} + N + 5},$$

$$c_1 = \frac{2}{(N + 1.3)^2 + \mu_{\text{eff}}},$$

and

$$c_w = \min\left(1 - c_1, \frac{2(\mu_{\text{eff}} + 1/\mu_{\text{eff}} - 2)}{(N + 2)^2 + \mu_{\text{eff}}}\right).$$

For the budget of the sum of function and constraint evaluations $10^5 N$ has been used.

4.4 Results

The results are presented by visualizing bootstrapped Empirical Cumulative Distribution Functions (ECDF). They show the percentages of function target values reached given a budget of function and constraint evaluations (shown on the x-axis normalized by the search space dimension and in logarithmic scale). The targets are defined as particular distances from the optimal value. In this work, the default COCO targets are used: $f_{\text{target}} = f_{\text{opt}} + 10^k$ for 51 different values of k between -8 and 2. Notice that COCO does not define any targets in the infeasible region. The crosses in the plots denote the medians of the sum of objective function and constraint evaluations (log10 of the medians is shown on the corresponding position on the x-axis) of the instances that were not able to reach all the targets. The crosses' values on the y-axis have no meaning, they are plotted on the particular line they belong to for better visualization purposes. Information about the experiments is shown (top left) as well as a legend (right).

Figure 1 shows the results as ECDF plots. The algorithm proposed in this work is denoted as "`maes on c`" in the plots. It is able to reach the most difficult target for all the dimensions shown.

Surprisingly, the iUDE, the LSHADE44, the ECHT-DE, and the ϵDEga are only able to reach the most difficult target for the smaller dimensions. One reason for this can stem from the way the equality constraint is turned into two inequality constraints using a tolerance of 10^{-8} (refer to inequalities (15) and (16)). To this end, further experiments have been performed (the results are provided in Appendix C) for those variants using a tolerance of 10^{-4} and keeping all other experimental setup considerations. It turns out from the corresponding ECDF plots that all variants except iUDE are still only able to reach the most difficult target for the smaller dimensions. The iUDE reached the most difficult target for all the dimensions considered. The performance of the other methods is better compared to the respective experiments with the tolerance of 10^{-8}.

The best performing variants are fmincon, the Active-Set ES, the MA-ES for general nonlinear equality constraints, the proposed MA-ES variant, the ϵMAg-ES, and the ConSaDE[4]. The performance of fmincon is not surprising, since the test problem is one where the underlying algorithms should perform well. A significant difference from the Active-Set ES and the MA-ES for general nonlinear equality constraints to the ES proposed in this work is that the former methods use repair for dealing with infeasible offspring. This is in contrast to the proposed closed form transformation in the developed method. The ϵMAg-ES is of different nature in that it also evaluates individuals in the infeasible region. As the results show, it is also effective on the test problem used.

It was also experimented with a back-calculation of the mutation vector of an individual after the nonlinear transformation that ensures the feasibility (results not shown here). It turned out that the back-calculation is disadvantageous in this case of a nonlinear transformation. The variant with back-calculation has only been able to reach the most difficult target for the smaller dimensions. The reason for this unexpected observation remains unclear and should deserve further investigations.

5 CONCLUSION

An algorithm based on the MA-ES has been designed with support for a parabolic/hyperbolic equality constraint. It is an interior point method and evaluates itself on the nonlinear constraint manifold.

The developed ES has been experimentally evaluated on a scalable test function and compared to other (black-box) optimization methods. The results show that it is an effective method for handling such constraints. In particular the comparison to fmincon, which is a method that is predestined for such problems, highlights the proposed method's competitiveness.

In future work, more experimental evaluation is of interest. Especially evaluating the proposed algorithm on real-world black-box problems with such constraints can reveal its power. The experimental results show that the Active-Set ES is an effective method for the considered problem as well. However, it is worth noting that it uses an iterative repair procedure for treating infeasible individuals. This is in contrast to the relatively simple and computationally cheap closed form transformation for ensuring feasibility presented in this work.

Another direction for future work could concern the considered DE methods except ConSaDE: It is of interest to understand why they do not reach the most difficult target, especially for the larger dimensions.

ACKNOWLEDGMENTS

This work was partially supported by the Austrian Science Fund FWF under grant P29651-N32.

APPENDIX

A PROOF OF THE TRANSFORMATION ENSURING FEASIBILITY

For showing that Equation (10) ensures feasibility, as a first step properties of $\mathbf{S}_-$, $\mathbf{S}_+$, $\mathbf{P}_-$, $\mathbf{P}_0$, and $\mathbf{P}_+$ are derived.

For $\mathbf{P}_-\mathbf{P}_-$, one can derive

$$\begin{aligned}\mathbf{P}_-\mathbf{P}_- &= \sum_{i:\lambda_i<0} \mathbf{u}_i\mathbf{u}_i^T \sum_{i':\lambda_{i'}<0} \mathbf{u}_{i'}\mathbf{u}_{i'}^T = \sum_{i:\lambda_i<0}\sum_{i':\lambda_{i'}<0} \mathbf{u}_i\mathbf{u}_i^T\mathbf{u}_{i'}\mathbf{u}_{i'}^T \\ &= \sum_{i:\lambda_i<0}\sum_{i':\lambda_{i'}<0} \mathbf{u}_i\delta_{ii'}\mathbf{u}_{i'}^T = \sum_{i:\lambda_i<0} \mathbf{u}_i\mathbf{u}_i^T = \mathbf{P}_-.\end{aligned} \tag{17}$$

Analogously, one gets

$$\mathbf{P}_+\mathbf{P}_+ = \mathbf{P}_+ \tag{18}$$

and similarly

$$\mathbf{P}_+\mathbf{P}_- = \mathbf{0}, \tag{19}$$

$$\mathbf{P}_-\mathbf{P}_+ = \mathbf{0}, \tag{20}$$

[4]It is worth noting that ConSaDE uses fmincon internally after a predefined number of generations to speed up the convergence.

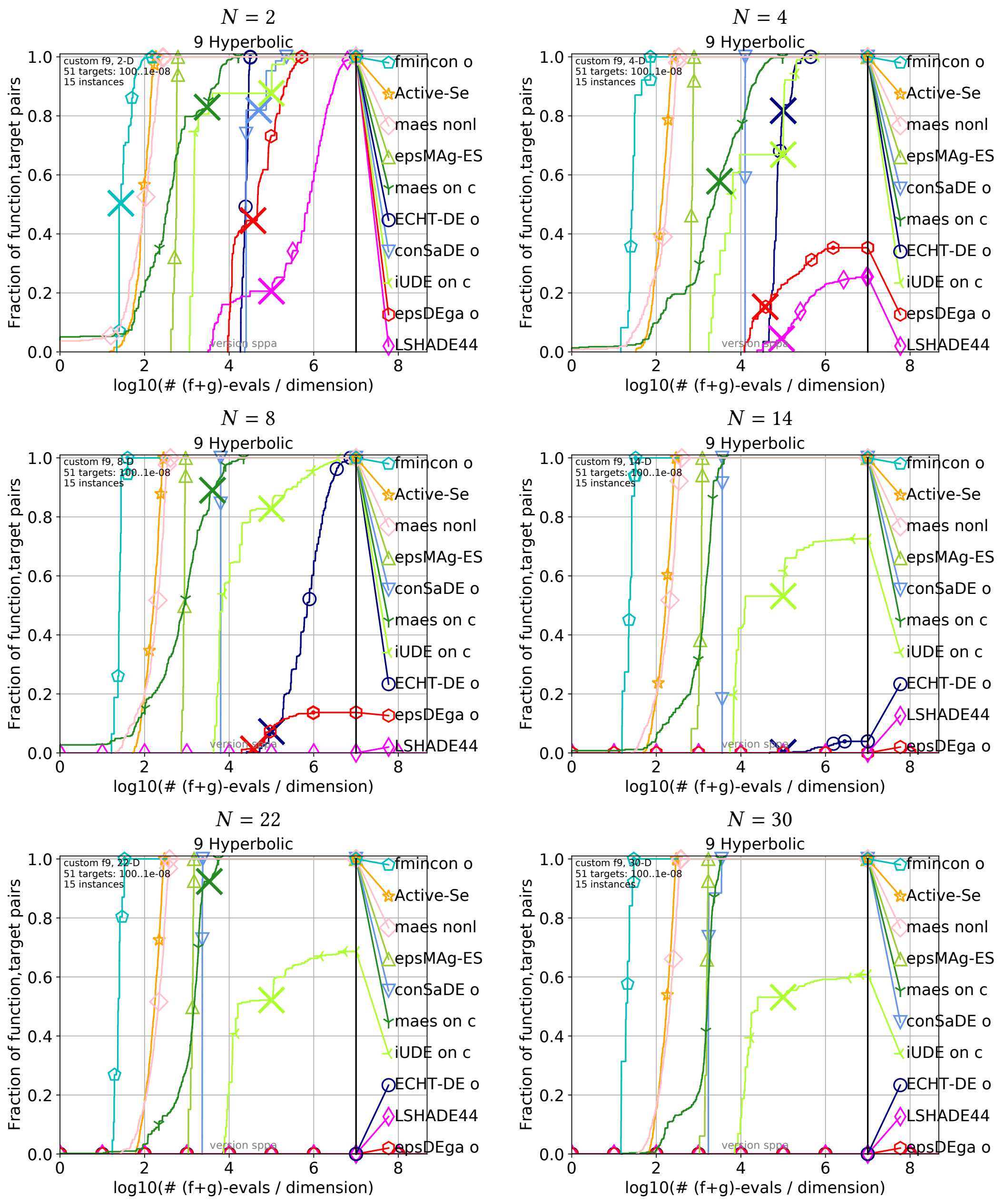


Figure 1: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of the various algorithms.

and

$$\mathbf{P}_-\mathbf{P}_0 = \mathbf{P}_0\mathbf{P}_- = \mathbf{P}_+\mathbf{P}_0 = \mathbf{P}_0\mathbf{P}_+ = \mathbf{0} \tag{21}$$

can be derived. For $\mathbf{P}_-^T$, one obtains

$$\mathbf{P}_-^T = \left(\sum_{i:\lambda_i<0} \mathbf{u}_i\mathbf{u}_i^T\right)^T = \sum_{i:\lambda_i<0} \left(\mathbf{u}_i\mathbf{u}_i^T\right)^T = \sum_{i:\lambda_i<0} \mathbf{u}_i\mathbf{u}_i^T = \mathbf{P}_-. \tag{22}$$

The derivation for

$$\mathbf{P}_+^T = \mathbf{P}_+ \tag{23}$$

is analogous.

For $\mathbf{S}_-\mathbf{P}_-$, one can derive

$$\begin{aligned}\mathbf{S}_-\mathbf{P}_- &= \sum_{i:\lambda_i<0} \lambda_i\mathbf{u}_i\mathbf{u}_i^T \sum_{i':\lambda_{i'}<0} \mathbf{u}_{i'}\mathbf{u}_{i'}^T = \sum_{i:\lambda_i<0}\sum_{i':\lambda_{i'}<0} \lambda_i\mathbf{u}_i\mathbf{u}_i^T\mathbf{u}_{i'}\mathbf{u}_{i'}^T \\ &= \sum_{i:\lambda_i<0}\sum_{i':\lambda_{i'}<0} \lambda_i\mathbf{u}_i\delta_{ii'}\mathbf{u}_{i'}^T = \sum_{i:\lambda_i<0} \lambda_i\mathbf{u}_i\mathbf{u}_i^T = \mathbf{S}_-.\end{aligned} \tag{24}$$

Analogously, one gets

$$\mathbf{S}_+\mathbf{P}_+ = \mathbf{S}_+ \tag{25}$$

and similarly

$$\mathbf{S}_+\mathbf{P}_- = \mathbf{S}_+\mathbf{P}_0 = \mathbf{0} \tag{26}$$

and

$$\mathbf{S}_-\mathbf{P}_+ = \mathbf{S}_-\mathbf{P}_0 = \mathbf{0} \tag{27}$$

can be derived.

Now, use of Equation (10) results in

$$\mathbf{S}\tilde{\mathbf{x}} = \frac{\sqrt{\varkappa_-}}{\sqrt{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}}\mathbf{S}\mathbf{P}_-\mathbf{x} + \mathbf{P}_0\mathbf{x} + \frac{\sqrt{\varkappa_+}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{S}\mathbf{P}_+\mathbf{x}. \tag{28}$$

Due to (8), (24), (25), (26), and (27), $\mathbf{S}\mathbf{P}_- = (\mathbf{S}_- + \mathbf{S}_+)\mathbf{P}_- = \mathbf{S}_-$, $\mathbf{S}\mathbf{P}_+ = (\mathbf{S}_- + \mathbf{S}_+)\mathbf{P}_+ = \mathbf{S}_+$, and $\mathbf{S}\mathbf{P}_0 = (\mathbf{S}_- + \mathbf{S}_+)\mathbf{P}_0 = \mathbf{0}$ hold. Consequently,

$$\mathbf{S}\tilde{\mathbf{x}} = \frac{\sqrt{\varkappa_-}}{\sqrt{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}}\mathbf{S}_-\mathbf{x} + \frac{\sqrt{\varkappa_+}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{S}_+\mathbf{x}. \tag{29}$$

Using (10), (22), and (23), one can derive

$$\tilde{\mathbf{x}}^T\mathbf{S}\tilde{\mathbf{x}} = \mathbf{x}^T\left[\frac{\sqrt{\varkappa_-}}{\sqrt{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}}\mathbf{P}_- + \frac{\sqrt{\varkappa_+}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{P}_+\right]\left[\frac{\sqrt{\varkappa_-}}{\sqrt{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}}\mathbf{S}_-\mathbf{x} + \frac{\sqrt{\varkappa_+}}{\sqrt{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}}\mathbf{S}_+\mathbf{x}\right]. \tag{30}$$

This can further be rewritten using (24), (25), (26), and (27) yielding

$$\frac{\varkappa_-}{-\mathbf{x}^T\mathbf{S}_-\mathbf{x}}\mathbf{x}^T\mathbf{S}_-\mathbf{x}+\mathbf{x}^T\mathbf{0}\mathbf{x}+\mathbf{x}^T\mathbf{0}\mathbf{x}+\frac{\varkappa_+}{\mathbf{x}^T\mathbf{S}_+\mathbf{x}}\mathbf{x}^T\mathbf{S}_+\mathbf{x} = -\varkappa_-+\varkappa_+ = \varkappa, \tag{31}$$

where the last step is justified by (7). □

B TWO-DIMENSIONAL EXAMPLES OF THE PROBLEM

Examples of the considered problem in two dimensions are shown in Figure 2:

- The top plot shows a visualization of the problem

$$\begin{aligned}&\min.\ ((\mathbf{x})_1 - 1)^2 + (\mathbf{x})_2^2\\ &\text{s.t. } \mathbf{x}^T\begin{pmatrix}1.0 & 0.1\\ 0.2 & 2.0\end{pmatrix}\mathbf{x} = 1.0.\end{aligned} \tag{32}$$

- The middle plot shows a visualization of the problem

$$\begin{aligned}&\min.\ ((\mathbf{x})_1 - 1)^2 + (\mathbf{x})_2^2\\ &\text{s.t. } \mathbf{x}^T\begin{pmatrix}1.0 & 0.5\\ 1.0 & -1.0\end{pmatrix}\mathbf{x} = 1.0.\end{aligned} \tag{33}$$

- The bottom plot shows a visualization of the problem

$$\begin{aligned}&\min.\ ((\mathbf{x})_1 - 1)^2 + (\mathbf{x})_2^2\\ &\text{s.t. } \mathbf{x}^T\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\mathbf{x} = 1.0.\end{aligned} \tag{34}$$

C DE EVALUATION RESULTS WITH CONSTRAINT TOLERANCE 10^{-4}

As shown in the paper, the iUDE, the LSHADE44, the ECHT-DE, and the ϵDEga are only able to reach the most difficult target for the smaller dimensions. One reason for this can stem from the way the equality constraint is turned into two inequality constraints using a tolerance of 10^{-8}. To investigate this further, experiments have been performed for those variants using a tolerance of 10^{-4} and keeping all other experimental setup considerations. The corresponding ECDF plots are shown in Figure 3 and Figure 4. All variants except iUDE are still only able to reach the most difficult target for the smaller dimensions. However, note that their performance is better compared to the experiments with the tolerance of 10^{-8}.

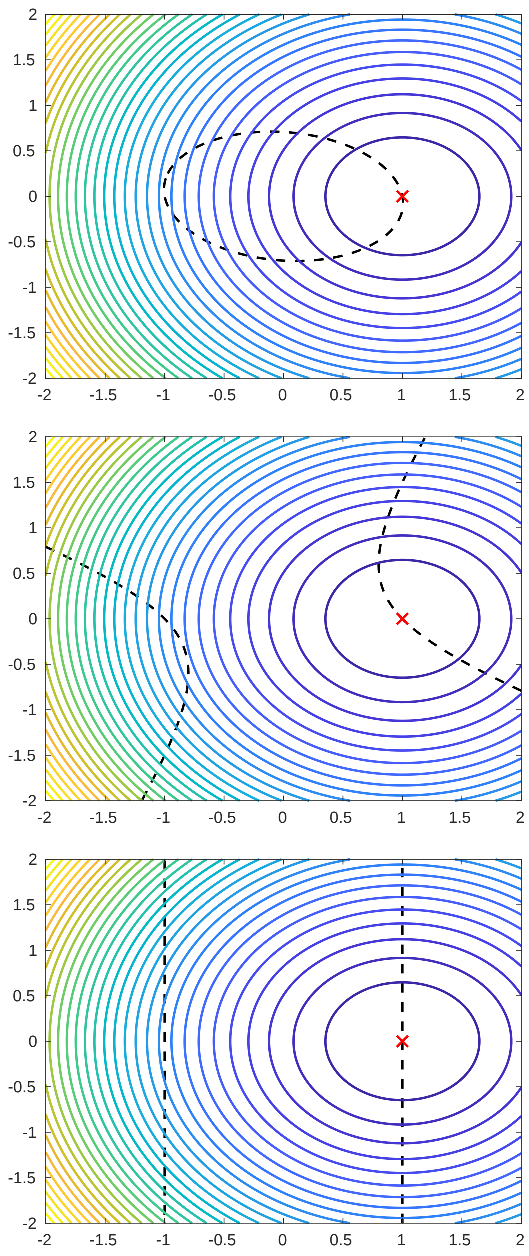


Figure 2: Two-dimensional visualization of a spherical problem (solid contour lines are shown) with an elliptical (top), a hyperbolic (middle), and a parabolic (bottom) equality constraint (shown as black dashed lines).

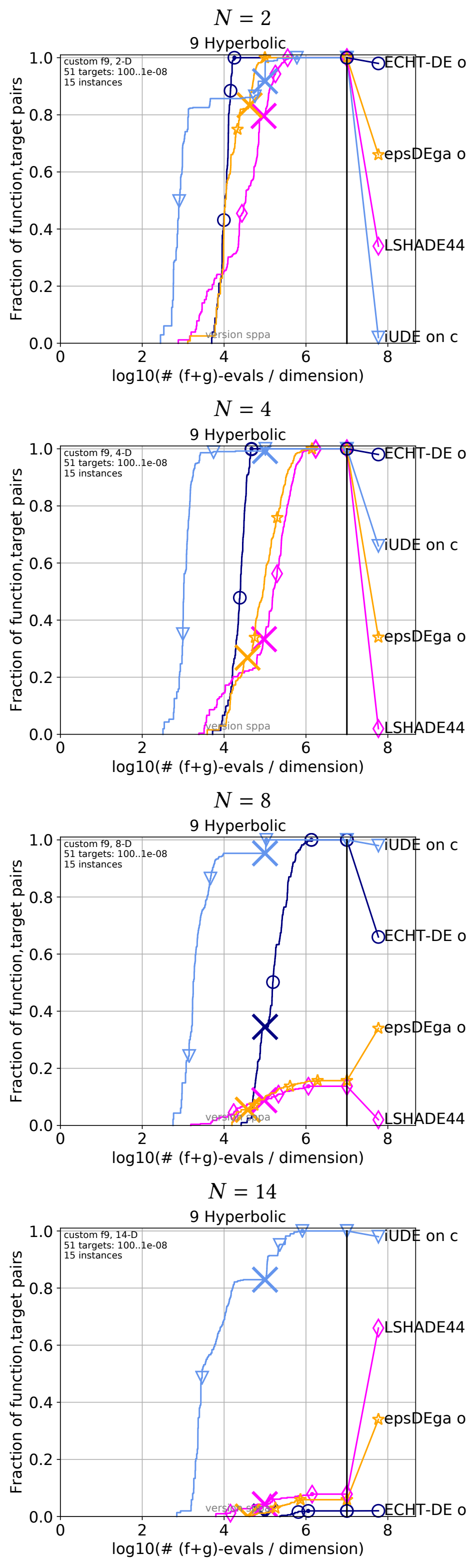


Figure 3: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of DE algorithms with an equality constraint tolerance of 10^{-4}. (Part 1/2)

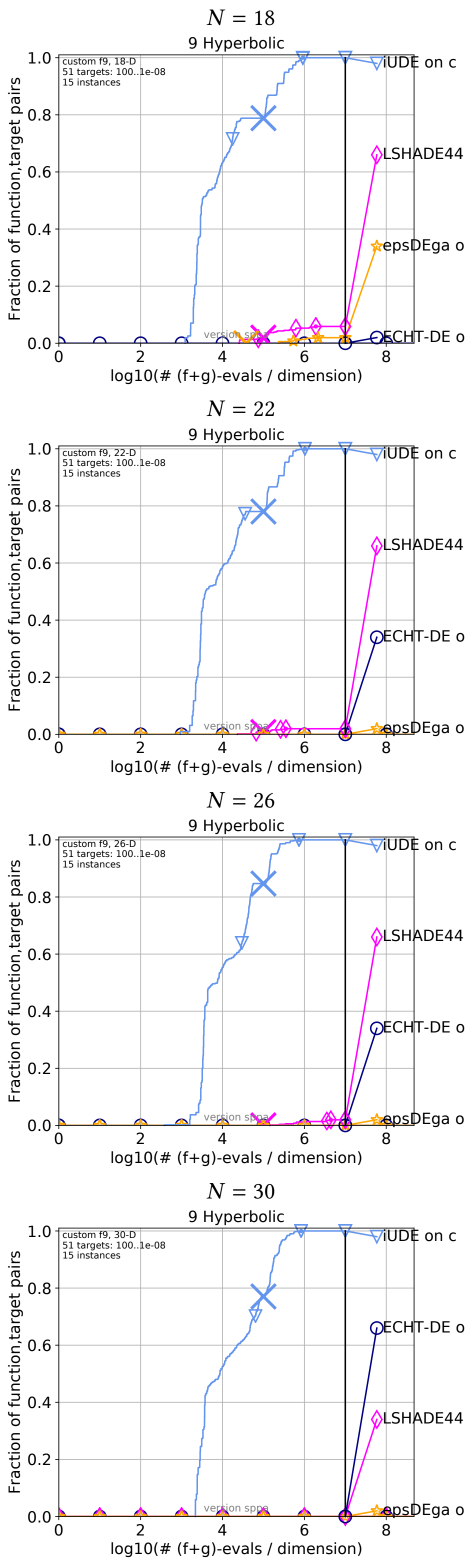


Figure 4: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of DE algorithms with an equality constraint tolerance of 10^{-4}. (Part 2/2)

REFERENCES

[1] Dirk V. Arnold. 2016. An Active-Set Evolution Strategy for Optimization with Known Constraints. In *International Conference on Parallel Problem Solving from Nature*. Springer, 192–202.

[2] Hans-Georg Beyer and Bernhard Sendhoff. 2017. Simplify Your Covariance Matrix Adaptation Evolution Strategy. *IEEE Transactions on Evolutionary Computation* 21, 5 (Oct 2017), 746–759.

[3] Steffen Finck. 2019. Worst Case Search over a Set of Forecasting Scenarios Applied to Financial Stress-Testing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. ACM, New York, NY, USA, 1722–1730.

[4] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2017. *COCO Documentation, Release 15.03*. http://coco.gforge.inria.fr/

[5] Michael Hellwig and Hans-Georg Beyer. 2018. A Matrix Adaptation Evolution Strategy for Constrained Real-Parameter Optimization. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (2018-10-04). IEEE, Rio de Janeiro, Brazil, Brazil, 1–8. https://doi.org/10.1109/CEC.2018.8477950

[6] Vicky Ling Huang, A Kai Qin, and Ponnuthurai N Suganthan. 2006. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 17–24.

[7] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. 2010. Differential Evolution with Ensemble of Constraint Handling Techniques for Solving CEC 2010 Benchmark Problems. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[8] Efrén Mezura-Montes and Carlos A. Coello Coello. 2011. Constraint-Handling in Nature-inspired Numerical Optimization: Past, Present and Future. *Swarm and Evolutionary Computation* 1, 4 (2011), 173–194.

[9] R. Poláková. 2017. L-SHADE with competing strategies applied to constrained optimization. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 1683–1689. https://doi.org/10.1109/CEC.2017.7969504

[10] Patrick Spettel and Hans-Georg Beyer. 2019. Matrix Adaptation Evolution Strategies for Optimization Under Nonlinear Equality Constraints. *Swarm and Evolutionary Computation* (2019). accepted.

[11] Patrick Spettel, Hans-Georg Beyer, and Michael Hellwig. 2019. A Covariance Matrix Self-Adaptation Evolution Strategy for Optimization Under Linear Constraints. *IEEE Transactions on Evolutionary Computation* 23, 3 (June 2019), 514–524. https://doi.org/10.1109/TEVC.2018.2871944 © 2019 IEEE.

[12] Tetsuyuki Takahama and Setsuko Sakai. 2010. Constrained Optimization by the ε Constrained Differential Evolution with an Archive and Gradient-Based Mutation. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–9.

[13] Ryoji Tanabe and Alex S. Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *IEEE Congress on Evolutionary Computation (CEC)*. 1658–1665. https://doi.org/10.1109/CEC.2014.6900380

[14] Joseph J. Thomson. 1904. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 7, 39 (1904), 237–265. https://doi.org/10.1080/14786440409463107

[15] Anupam Trivedi, Krishnendu Sanyal, Pranjal Verma, and Dipti Srinivasan. 2017. A unified differential evolution algorithm for constrained optimization problems. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 1231–1238. https://doi.org/10.1109/CEC.2017.7969446

A Matrix Adaptation Evolution Strategy for Optimization on General Quadratic Manifolds

Patrick Spettel and Hans-Georg Beyer

Supplementary Material

D FURTHER EXPERIMENTAL EVALUATION RESULTS

For the quantitative evaluation, problems are generated by randomly sampling N eigenvalues $\lambda_j \sim \mathcal{N}(0,1)$ for $j \in \{1,\dots,N\}$ resulting in a quadratic constraint. Using those eigenvalues, a randomized $\mathbf{S}$ matrix is computed with $\mathbf{S} = \mathbf{U}^T\mathbf{D}\mathbf{U}$, where $\mathbf{U}$ is also randomly generated such that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{D} = \operatorname{diag}(\lambda_1,\dots,\lambda_N)$. To ensure that there are not only hyperbolically constrained problems, additional problems are considered: Absolute eigenvalues are considered to have additional elliptically constrained problems. In addition, half of the absolute eigenvalues are set to 0 to generate additional parabolic problems. For the objective function, a randomized ellipsoid function

$$f(\mathbf{x}) = \sum_{i=1}^{N} 10^{\frac{i-1}{N-1}} \left(\mathbf{R}\mathbf{x} + \mathbf{t}\right)_i^2 \tag{35}$$

is used, where $\mathbf{R}_{jk} \sim \mathcal{N}(0,1)$ and $\mathbf{t}_j \sim \mathcal{N}(0,1)$. ECDF plots are shown in Figure 5 for dimension 10, Figure 6 for dimension 20, Figure 7 for dimension 30, and Figure 8 for dimension 40. The optimal values used as a reference for creating the ECDF plots are determined by solving the randomly generated optimization problems using fmincon.

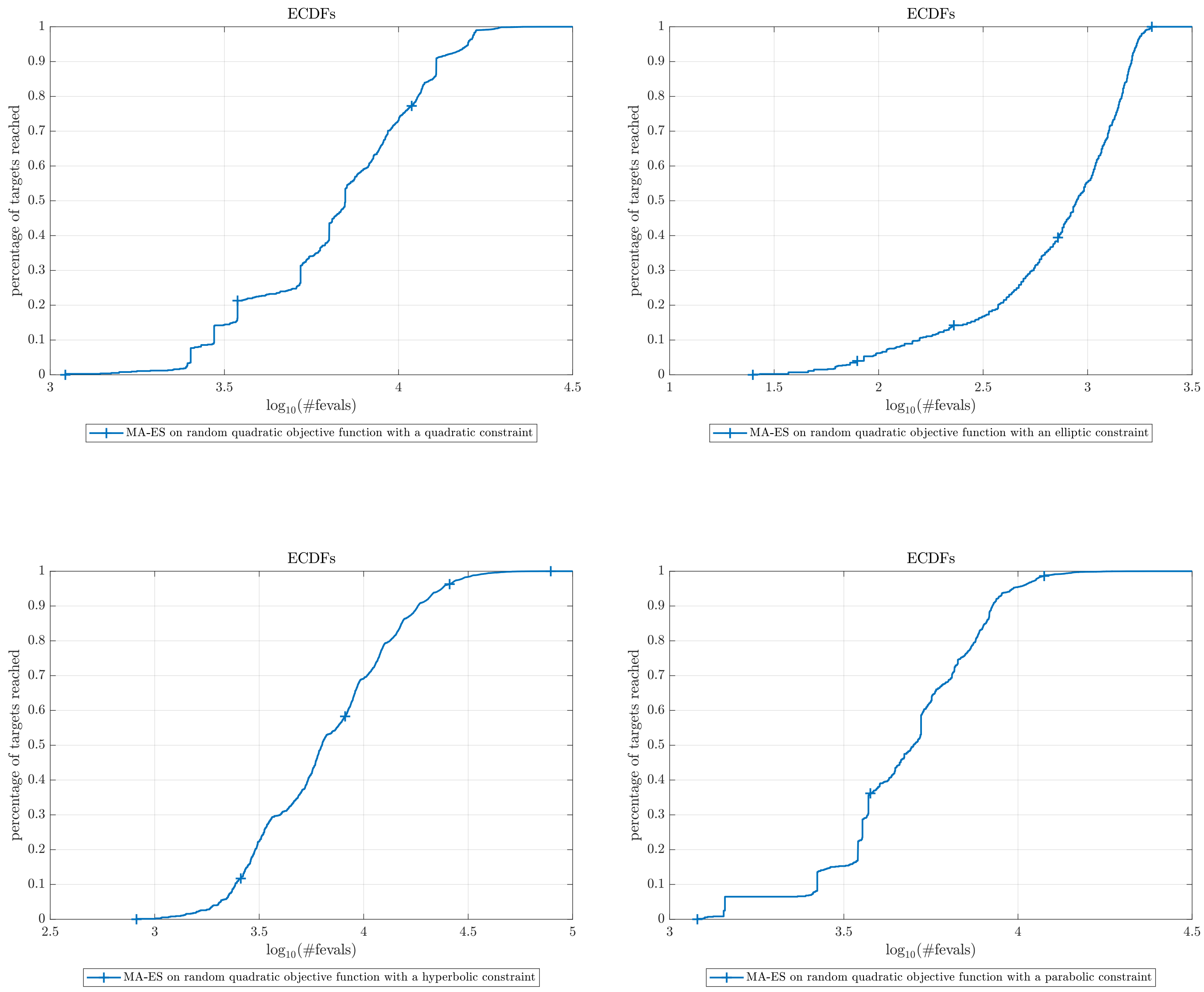


Figure 5: Results (as ECDF plots) of applying the proposed algorithm to the randomly generated problems for dimension 10.

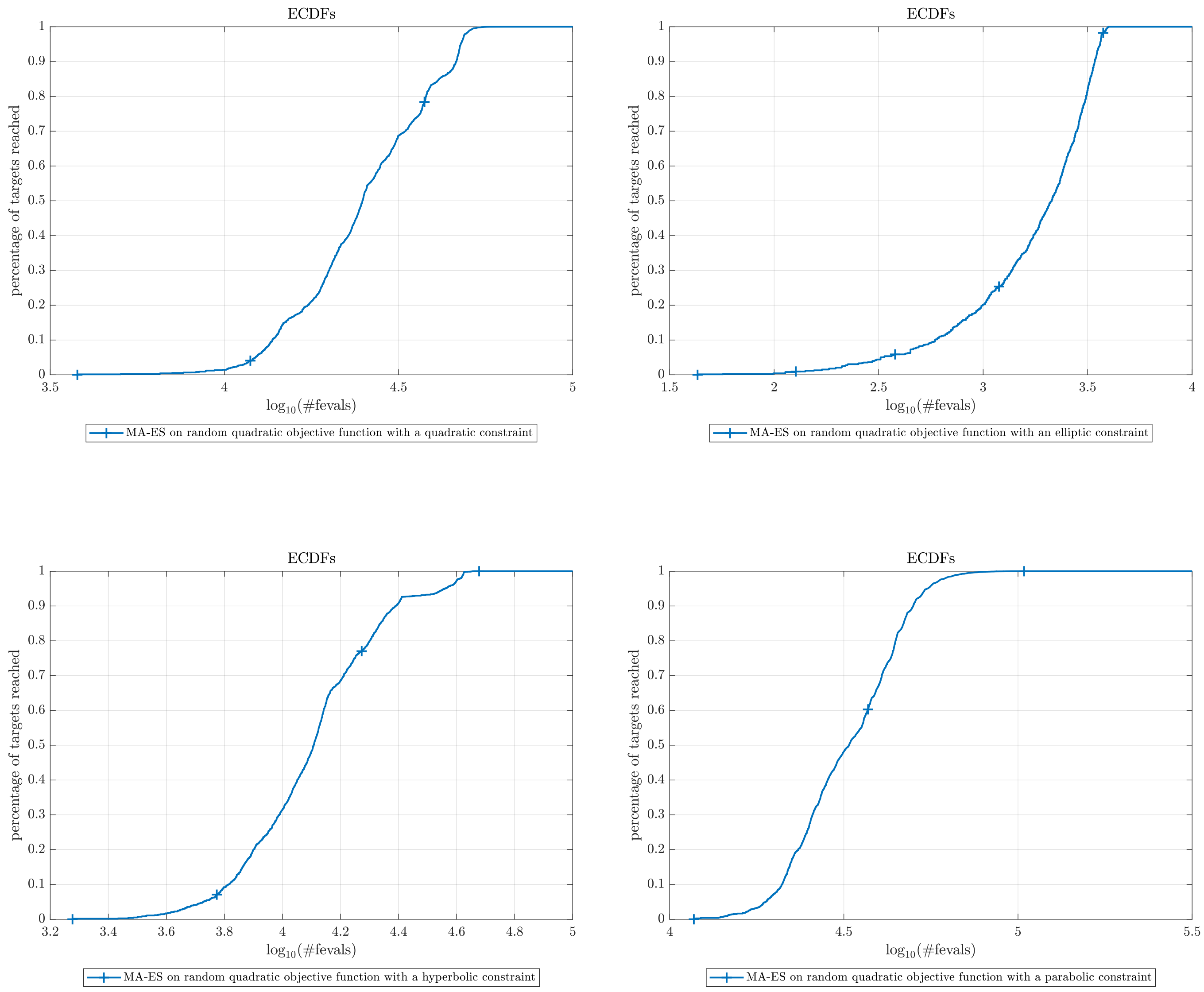
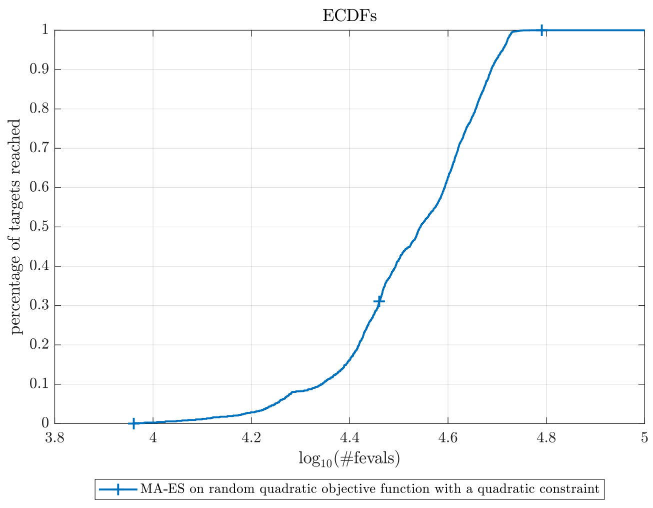
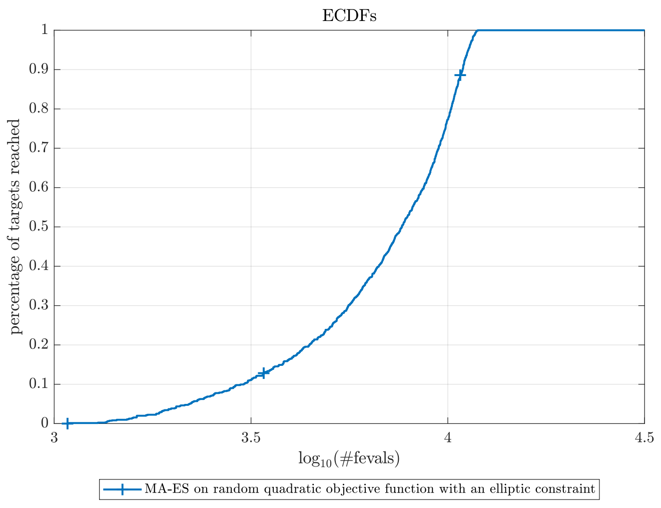
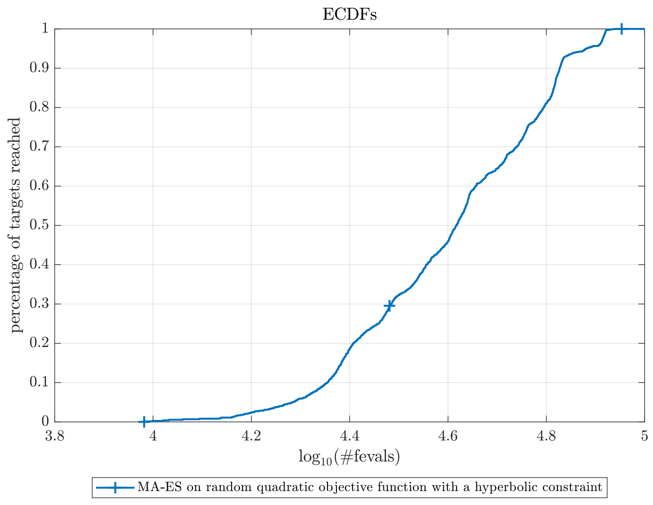


Figure 6: Results (as ECDF plots) of applying the proposed algorithm to the randomly generated problems for dimension 20**.**







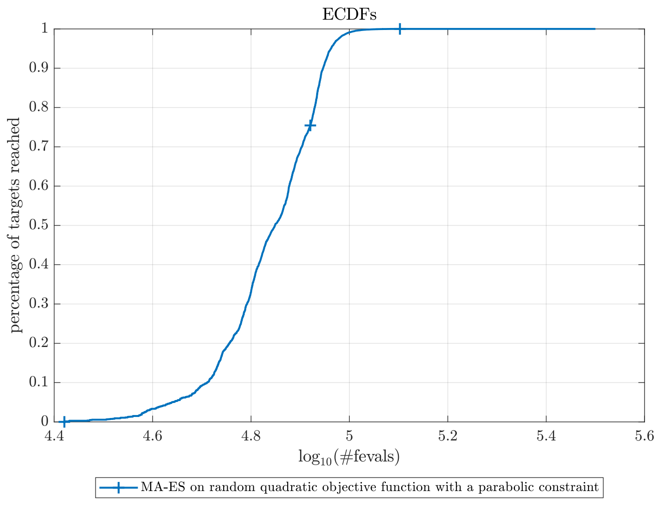


Figure 7: Results (as ECDF plots) of applying the proposed algorithm to the randomly generated problems for dimension 30**.**

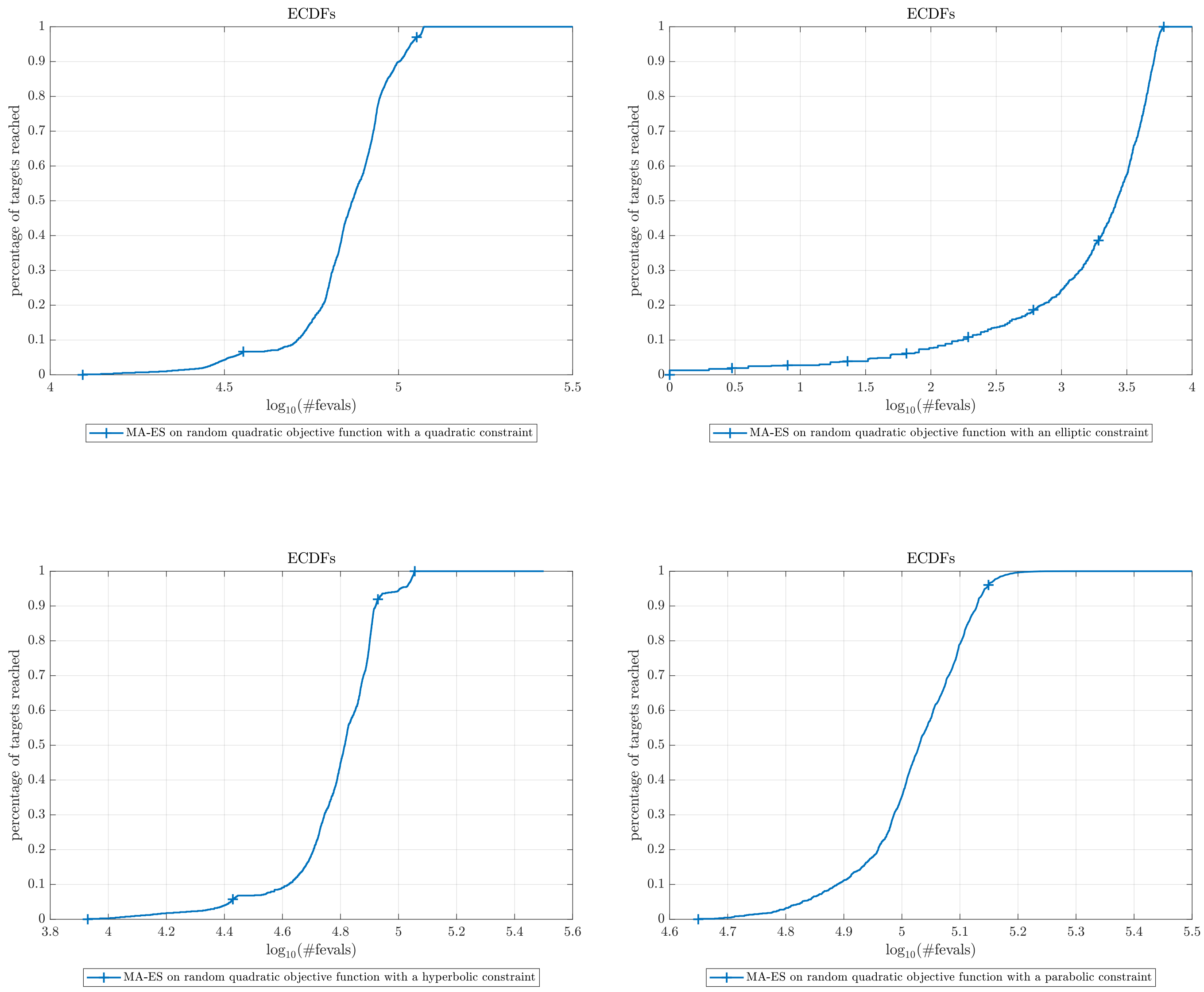


Figure 8: Results (as ECDF plots) of applying the proposed algorithm to the randomly generated problems for dimension 40**.**