technische universität
dortmund

Constraint Handling for Covariance Matrix Adaption with Meta-Models

Oliver Kramer
Günter Rudolph
André Barthelmes

Algorithm Engineering Report
TR09-2-004
Feb. 2009
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

Constraint Handling for Covariance Matrix Adaptation with Meta-Models

Oliver Kramer oliver.kramer@tu-dortmund.de
Department of Computer Science, Technische Universität Dortmund, 44227 Dortmund, Germany

Günter Rudolph guenter.rudolph@tu-dortmund.de
Department of Computer Science, Technische Universität Dortmund, 44227 Dortmund, Germany

André Barthelmes andre.barthelmes@tu-dortmund.de
Department of Computer Science, Technische Universität Dortmund, 44227 Dortmund, Germany

Abstract
Evolution strategies are successful black-box optimization methods. But many practical numerical problems are constrained. Whenever the optimum lies in the vicinity of the constraint boundary, the success rates decrease and make successful random mutations almost impossible. Low success rates result in premature step size reduction and finally lead to premature convergence. We try to draw a consistent picture of recent developments of heuristics that try to increase the success rate situation at the constraint boundary – from simple death penalty of evolution strategies to covariance matrix adaptation. Meta-models of the constraint functions turn out to be a promising approach for savings in computation time both in terms of fitness and constraint function evaluations. We use meta-models to rotate the mutation ellipsoid, to check the feasibility of candidate solutions, and to repair infeasible mutations at the constraint boundary.

Keywords
evolution strategies, covariance matrix adaptation, constraint handling, meta-modeling, repair algorithm

1 Introduction

Many continuous optimization problems in practical applications are subject to constraints [7]. Constraints can make an easy problem hard and hard problems even harder. Surprisingly, in the past only little research efforts have been devoted to the development of efficient and effective constraint handling techniques – in contrast to the energy invested in the development of new methods for unconstrained optimization. This observation also holds true in the field of natural and evolutionary computation.

This paper summarizes our research results regarding constraint handling within evolution strategies (ES). In general, the constrained continuous nonlinear programming problem is defined as follows: Find a solution $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ in the N-dimensional solution space $\mathbb{N}^N$ that minimizes the objective function $f(\mathbf{x})$; in symbols:

$$\begin{array}{lll} f(\mathbf{x}) \rightarrow \min!, & \mathbf{x} \in \mathbb{N}^N & \text{subject to} \\ \text{inequalities} & g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, n_1, \text{ and} \\ \text{equalities} & h_j(\mathbf{x}) = 0, & j = 1, \ldots, n_2\,. \end{array} \tag{1}$$

A feasible solution $\mathbf{x} \in \mathbb{N}^N$ satisfies all n_1 inequality and n_2 equality constraints. A feasible solution that minimizes $f(\cdot)$ is termed an optimal solution. If $g_i(\mathbf{x}^*) = 0$ for some inequality constraint at an optimal solution $\mathbf{x}^*$ then the constraint is said to be active.

In the field of evolutionary computation the constraints typically are not considered available in their explicit formal form. Rather, the constraints are assumed to be black boxes: a vector $\mathbf{x}$ fed to the black box just returns a numerical or boolean value. If there is a numerical response then the information about a positive value can be used to assess the distance to feasible solutions. A number of constraint handling methods exploit this information. Our contributions to this line of research is summarized in section 2. If the constraint function only provides information whether the solution is feasible or not then other constraint handling techniques must be deployed. This situation is the starting point of our approach described in sections 3 and 4. We assume that the evaluations of the constraint functions are computationally expensive and that the return values are boolean. The resulting constraint handling method stems from the idea of building a meta-model of the constraint functions such that we can make use of more powerful constraint handling methods only applicable if constraints are explicitly given. Apart from obtaining more effective constraint handling methods the meta-model approach also yields savings in computation time, provided the scenario assumed here holds true.

2 Constraint Handling Methods

A variety of constraint-handling methods for evolutionary algorithms have been developed in the last decades. Most of them can be classified into five main types of concepts.

- *Penalty functions* decrease the fitness of infeasible solutions by taking the number of infeasible constraints or the distance to feasibility into account [1, 19, 10, 12, 6].
- *Repair algorithms* either replace infeasible solutions or only use the repaired solutions for evaluation of their infeasible pendants [2, 5].
- *Decoder functions* map genotypes to phenotypes which are guaranteed to be feasible. Decoders build up a relationship between the constrained solution space and an artificial solution space easier to handle [5, 13, 21].
- *Feasibility preserving representations and operators* force candidate solutions to be feasible [23, 25].
- *Multiobjective optimization* techniques are based on the idea of handling each constraint as an objective [4, 11, 24, 26].

The easiest and most frequently used procedure to handle constraints is discarding infeasible solutions within an iteration loop until λ feasible offspring solutions exist. This procedure is known as *death penalty*. Death penalty is no satisfying constraint handling method. Nevertheless, the analysis of death penalty delivers valuable insights into the situation at the constraint boundary: Stochastic optimization algorithms on constrained optimization problems suffer from premature step size reduction in case of active inequality constraints. This results in premature convergence. Broadly speaking, the reason for the premature step size reduction is the fact that the constrained

region cuts off the mutative success area and decreases the success rate significantly. Low success rate prevent the production of successful solutions or force self-adaptive step sizes to decrease.

To get an impression of the situation at the constraint boundary and the behavior of methods to tackle premature convergence, we repeat their basic principles, state their experimental behavior on two relevant test problems and summarize their advantages and disadvantages. The two test functions excellently demonstrate the phenomenon of premature fitness stagnation. Problem 2.40 – taken from Schwefel's artificial test problems [3] – exhibits a linear objective function and an optimum with five active linear constraints. The problem is to minimize

$$f_{2.40}(\vec{x}) = -\sum_{i=1}^{5} x_i \tag{2}$$

subject to

$$g_{2.40,j}(\vec{x}) = \begin{cases} x_j \geq 0, & \text{for } j = 1, \ldots, 5 \\ -\sum_{i=1}^{5}(9+i)x_i + 50000 \geq 0, & \text{for } j = 6 \end{cases} \tag{3}$$

with minimum $\vec{x}^* = (5000, 0, 0, 0, 0)^T$ and $f(\vec{x}^*) = -5000$. Problem TR (tangent problem) is based on the sphere model subject to one linear constraint

$$f_{\text{TR}}(\vec{x}) = \sum_{i=1}^{N} x_i \quad \text{with} \quad g_{\text{TR}}(\vec{x}) = \sum x_i - N > 0 \tag{4}$$

with $\vec{x}^* = (1, \ldots, 1)^T$ and $f(\vec{x}^*) = N$. The success rates on TR get worse when approximating the optimum.

2.1 Death Penalty

First of all, we will analyze the behavior of death penalty, i.e. simply discarding infeasible offspring solutions. This is the first time we can observe premature fitness stagnation. Figure 1 shows the corresponding result of a (15,100)-ES with the following settings on problems 2.40 and TR2.

Experimental settings	
Population model	(15,100)
Mutation type	standard, $n_\sigma = N$, $\tau_0 = (\sqrt{2n})^{-1}$ and $\tau_1 = (\sqrt{2\sqrt{n}})^{-1}$
Crossover type	intermediate, $\rho = 2$
Selection type	comma
Initialization	$\sigma_i = \frac{\lvert x^{(0)} - x^* \rvert}{N}$
Termination	fitness stagnation, $\theta = 10^{-12}$
Runs	25

Particularly, we make use of self-adaptive Gaussian mutation within evolution strategies

$$\vec{x}' = \vec{x} + (\sigma_1 \mathcal{N}_1(0,1), \ldots, \sigma_N \mathcal{N}_N(0,1)) \tag{5}$$

with step sizes σ_i, $1 \leq i \leq N$. For a comprehensive introduction to evolution strategies and their mutation operators, we refer to the introduction of Beyer and Schwefel [3].

For an introduction to self-adaptation we refer to [15]. All experiments in this article make use of the same experimental settings unless stated explicitly. The termination condition is fitness stagnation: the algorithms terminate if the fitness win from generation t to generation $t+1$ falls below $\theta = 10^{-12}$. In this case the magnitude of the step sizes is too small to effect further improvements. Parameters *best*, *mean*, *worst* and *dev* describe the achieved fitness[1] of 25 experimental runs while *ffe* counts the average number of fitness function evaluations and *cfe* of constraint function evaluations respectively. The results show that death penalty is not able to approximate the optimum of the problem satisfactorily. The relatively high standard deviations *dev* show that the algorithms produce unsatisfactorily different results. Under simple conditions, i.e. a linear objective function, linear constraints and a comparably simple mutation operator, the occurrence of premature convergence due to a premature decrease of step sizes was proved [14].

Death Penalty	best	mean	worst	dev	ffe	cfe
TR2	$4.1 \cdot 10^{-7}$	$3.1 \cdot 10^{-4}$	$1.4 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$	11,720	20,447
2.40	51.9	227.6	390.0	65.2	50,624	96,817

Table 1: Experimental results of the death penalty method.

We can summarize the behavior of death penalty with the advantages:

- *Death penalty* is easy to implement.

The disadvantages are:

- *Death penalty* is inefficient as many infeasible tries are wasted,
- it suffers from premature convergence.

The following methods aim at preventing premature convergence.

2.2 Dynamic Penalty Function

The question arises whether dynamic penalty functions also suffer from premature convergence. To answer this question we tested the penalty function by Joines and Houck [12] that is based on adding a penalty on infeasible solutions

$$\tilde{f}(\vec{x}) = f(\vec{x})(C \cdot t)^{\alpha} \cdot \sum_{i=1}^{n_1} G_i^{\beta}. \tag{6}$$

with parameters C, α, β and the constraint violation G_i. In particular, the penalty depends on the number of iterations t and decreases in the course of time. Table 2 shows the experimental analysis of the penalty function on 2.40 and TR2 with $\alpha = 1.0$ and $\beta = 1.0$. Again, the algorithm is based on a (15,100)-ES with the same settings like in the last paragraph 2.1. The algorithm stops earlier, but the results are even worse and show that premature fitness stagnation occurs, too. The reason is quite obvious: the success rate in the vicinity of the infeasible search space remains small because of the penalty − no matter if caused by discarding or penalizing. Consequently, we can summarize:

[1] difference between the optimum and the best solution $|f(\vec{x}^*) - f(\vec{x}^{best})|$

- *Dynamic penalty functions* are easy to implement, and
- no feasible starting point is required.

But the disadvantages are:

- *Dynamic penalty functions* suffer from premature convergence.

	best	mean	worst	dev	ffe	cfe
TR2	$1.2 \cdot 10^{-6}$	$1.2 \cdot 10^{-3}$	$6.7 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	13,100	13,100
2.40	219.4	440.8	641.5	85.0	31,878	31,878

Table 2: Experimental results of the dynamic penalty function by Joines and Houck [12] on problems TR2 and 2.40.

2.3 Death Penalty Step Control

The most obvious modification to prevent premature step size reduction is the introduction of a minimum step size ϵ that forces all step sizes σ_i with $1 \leq i \leq N$ to maintain

$$\sigma_i \geq \epsilon. \tag{7}$$

This is exactly what the death penalty step control evolution strategy (DSES) is aiming at [17]. Nevertheless, a lower bound on the step sizes will also prevent an unlimited approximation of the optimum when reaching the range of ϵ. Consequently, the DSES makes use of a control mechanism to reduce ϵ during convergence to the optimal solution. The reduction process depends on the number of infeasible mutations produced when reaching the area of the optimum at the boundary of the feasible solution space. The reduction process of ϵ depends on the number z of rejected infeasible solutions: Every ϖ infeasible trials, ϵ is reduced by a factor $0 < \vartheta < 1$ according to the equation

$$\epsilon' := \epsilon \cdot \vartheta. \tag{8}$$

The DSES is denoted by $[\varpi; \vartheta]$-DSES. Again, we show the behavior of the constraint handling method on problem TR2 and 2.40, see table 3. The method is able to approximate the optimum of problem 2.40 with comparably few fitness function evaluations, but a waste of constraint function evaluations. Intuitively, the five active linear constraints of problem 2.40 cause many infeasible samples, so does the step sizes reduction mechanism. On *harder* problems like TR2 the low success rates still prevent an arbitrarily exact approximation of the optimal solution. The success of the DSES depends on a proper *reduction speed*, i.e. proper parameter settings for ϵ and ϑ. Too fast reduction results in premature convergence, too slow reduction is inefficient. Further experiments on other test functions confirm this picture.

Again, we summarize the results:

- *Death penalty step control* is easy to implement, and
- shows and improvement of the approximation of optima with active constraints.

But the disadvantages are:

DSES	type	best	mean	dev	ffe	cfe
TR2	$[15; 0.5]$	$3.7 \cdot 10^{-9}$	$8.5 \cdot 10^{-6}$	$2.5 \cdot 10^{-6}$	1,253,394	2,315,574
2.40	$[100; 0.7]$	$1.9 \cdot 10^{-11}$	$2.7 \cdot 10^{-10}$	$7.9 \cdot 10^{-10}$	89,832	1,118,490

Table 3: Experimental results of the DSES.

- *Death penalty step control* consumes many constraint function evaluations,
- its success depends on proper parameter settings, and
- on some problems it may still suffer from low success rates.

2.4 Biased Mutation

The shape of the standard mutation ellipsoid is Gaussian. The best modification to improve the success rate situation would be a more flexible mutation distribution function. Later, we will see that a rotation of the mutation ellipsoid − with reservations − is a reasonable undertaking. But is a deformation also an adequate solution to low success rates? Biased mutation aims at biasing the mean of the Gaussian distribution into beneficial directions self-adaptively [18]. A self-adaptive bias coefficient vector $\vec{\xi}$ determines the direction of this bias and augments the degree of freedom of the mutation operator. This additional degree of freedom improves the success rate of reproducing superior offspring. The mutation operators adapts the bias direction within the interval -1 (for left) and 1 (for right) in each of the N dimensions:

$$\vec{\xi} = (\xi_1, \ldots, \xi_N) \text{ with } -1 \leq \xi_i \leq 1. \tag{9}$$

This relative direction must be translated into an absolute bias vector. For this sake the step sizes σ_i can be used. For every $i \in 1, \ldots, N$ the bias vector $\vec{b} = (b_1, \ldots, b_N)$ is defined by:

$$b_i = \xi_i \cdot \sigma_i \tag{10}$$

Since the absolute value of bias coefficient ξ_i is less than or equal to 1, the bias will be bound to the step sizes σ_i. This restriction prevents the search from being biased too far away from the parent. Hence, the biased mutation works as follows:

$$\vec{x}' = \vec{x} + (\sigma_1 \mathcal{N}_1(0,1) + b_1, \ldots, \sigma_N \mathcal{N}_N(0,1) + b_N) \tag{11}$$

$$= \vec{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)) \tag{12}$$

To allow self-adaptation, the bias coefficients are mutated in the following *meta-EP* way:

$$\xi_i' = \xi_i + \gamma \cdot \mathcal{N}(0,1) \quad i = 1, \ldots, N. \tag{13}$$

with parameter γ determining the mutation strength on the bias. The biased mutation operator (BMO) biases the mean of mutation and enables the evolution strategy to reproduce offspring outside the standard mutation ellipsoid. To direct the search, the biased mutation enables the center of the ellipsoid to move within the bounds of the regular step sizes $\vec{\sigma}$. The bias moves the center of the Gaussian distribution within the bounds of the step sizes and improves the success rate situation as the success area increases. An *adaptive* variant of the originally *self-adaptive* biased mutation is the descent

mutation operator. It estimates the descent direction of two population's centers $\vec{\chi}_t$ and $\vec{\chi}_{t+1}$ of successive generations. Let $\vec{\chi}_t$ be the center of the population at generation t

$$\vec{\chi}_t = \sum_{i=1}^{\mu} \vec{x}_i \tag{14}$$

The normalized descent direction $\vec{\xi}$ of two successive population centers $\vec{\chi}_t$ and $\vec{\chi}_{t+1}$ is

$$\vec{\xi} = \frac{\vec{\chi}_{t+1} - \vec{\chi}_t}{|\vec{\chi}_{t+1} - \vec{\chi}_t|} \tag{15}$$

Similar to the BMO, the descent mutation operator (DMO) now becomes

$$\vec{x}' = \vec{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)) \tag{16}$$

The DMO is reasonable as long as the assumption of locality is true: the estimated direction of the global optimum can be derived from local information, i.e. the descent direction of two successive populations' centers. Again, we analyze both biased mutation operators on the test problems 2.40 and TR2 and show the results in table 4. For the sake of adaptation of the bias an increase of offspring individuals to $\lambda = 300$ is necessary. The bias mutation parameter is set to the standard setting $\gamma = 0.1$. Our experiments show that the BMO and the DMO are both able to improve the results on problem 2.40. The experiments reveal that the mutation distribution deformation improves the success rate – intuitively by shifting the center of the mutation ellipsoid so that the latter is not cut off by the infeasible solution space. But the results show that the *harder* problem TR2 is still not easy to approximate. We can conclude:

BMO	best	mean	dev	ffe	cfe
TR2	$1.6 \cdot 10^{-6}$	$9.0 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	26,832	25,479
2.40	$8.2 \cdot 10^{-12}$	$2.2 \cdot 10^{-7}$	$2.4 \cdot 10^{-8}$	459,774	508,387
DMO	best	mean	dev	ffe	cfe
TR2	$8.8 \cdot 10^{-9}$	$4.6 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	31,506	29,196
2.40	$1.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-9}$	$2.8 \cdot 10^{-10}$	358,954	359,545

Table 4: Experimental results of the biased mutation variants on TR2 and 2.40.

- *Biased mutation* improves the approximation of optima with active constraints.
- *Descent biased mutation* is comparatively efficient, in particular more efficient than the BMO.

But the disadvantages are:

- *Biased muation* consumes many fitness and constraint function evaluations, and
- on some problems it may still suffer from low success rates.

2.5 Two Sexes Evolution Strategy

Allowing infeasible solutions to cross the constraint boundary would improve the success rate. But how can we allow the constraint border crossing without penalizing infeasible mutations? A bio-inspired concept offers an answer to this question: The two sexes evolution strategy by Kramer and Schwefel [17] allows candidate solutions to cross the constraint boundary. The mechanism to enforce the approach of the optimum stems from nature. Individuals of different sex are selected by different criteria and nature allows pairing only between individuals of different sex. Transferring this principle to constraint handling means: Every individual of the two sexes evolution strategy (TSES) is assigned to a feature called sex. Similar to nature, individuals with different sexes are selected according to different criteria. Individuals with sex o are selected by the objective function. Individuals with sex c are selected by the fulfillment of constraints. The intermediary recombination operator plays a key role. Recombination is only allowed between parents of different sex. A few modifications are necessary to prevent an explosion of the step size, i.e. a two-step selection operator for individuals of sex c similar to the operator by Hoffmeister and Sprave [9]. For a list of TSES variants and modifications we refer to Kramer and Schwefel [17]. The populations are noted as $(\mu_o + \mu_c, \lambda_o + \lambda_c)$ − the index determines the sex, i.e. o for objective function and c for constraints.

TSES	type	κ	best	mean	dev	ffe / cfe
TR2	(8+8,10+90)	200	$5.4 \cdot 10^{-8}$	$2.9 \cdot 10^{-7}$	$4.7 \cdot 10^{-8}$	521,523
2.40	(8+8,13+87)	50	0.0	0.0	$3.7 \cdot 10^{-11}$	498,594

Table 5: Experimental results of the two sexes evolution strategy on TR2 and 2.40.

Table 5 shows the experimental results of the TSES on problems TR2 and 2.40. While death penalty completely fails on problem 2.40 the (8+8,13+87)-TSES reaches the optimum in every run. Now, a better approximation of the *harder* problem TR2 is possible. Nevertheless, the approximation quality may still be improved and an analysis on further test problems − that can be found in [16] − shows that the TSES is successful on many constrained problems, but not on all. Fortunately, the TSES is quite robust to the chosen population ratios. We can summarize that:

- The *two sexes evolution strategy* improves the approximation of optima with active constraints,
- allows infeasible starting points,
- saves constraint function evaluations, e.g. in comparison to the DSES, and
- is quite robust to parameter changes.

But the disadvantages are:

- The *two sexes evolution strategy* still consumes many fitness function evaluations,
- on some problems it may still suffer from low success rates.

No satisfying conclusion can be drawn. No constraint handling method is able to handle a harder problem like TR2 satisfactorily. Furthermore, the methods are still comparably inefficient.

3 Constraint Boundary Meta-Modeling

To construct more powerful constraint handling methods the idea arises to build a meta-model of the constraint function. This meta-model can be used in various kinds of ways. In the following we will use it for

1. Rotation of the mutation ellipsoid,
2. the check for feasibility of mutations, and
3. reparation of infeasible mutations.

The meta-model will be used for correlated mutations and for the covariance matrix adaptation evolution strategy.

3.1 A Linear Constraint Estimator

In a first step we concentrate on linear constraints. Among various linear classification algorithms like support vector machines or neural networks we developed a new efficient constraint boundary estimator based on fast binary search. We assume that our constraint handling method starts with the first occurrence of an infeasible individual. Our algorithms for the estimation of the linear constraint hyper-plane h works as follows. In the first step the meta-model estimator has to identify $N-1$ points *on* the N-dimensional linear constraint hyper-plane h. Our model-estimator makes use of an N-dimensional hypersphere that cuts the constraint boundary. The connection between the infeasible points on the hyper-sphere and its origin cuts the constraint boundary. The approach uses binary search to find the cutting points and works as follows:

1. Determination of the center point of the model-estimator: When the first infeasible offspring individual x_i is produced, the original feasible parent x_f is the center of the corresponding meta-model estimator.
2. Generation of random points on the surface of a hypersphere: Point x_f is the center of a hypersphere with radius r, such that the constraint boundary is cut. In N dimensions $N-1$ infeasible points $x_f^{(i)} \quad 1 \leq i \leq N-1$ have to be produced. Table 6 shows how many constraint function evaluations *cfe* are necessary in average until $N-1$ infeasible points are produced. The points on the surface are produced randomly with uniform distribution using the method of Marsaglia [20]. In the first step the algorithm produces N Gaussian distributed random numbers and scales the numbers to length 1. Further scaling and shifting yields N randomly distributed point on the hyper-sphere surface

$$x_i \sim \mathcal{N}(0,1) \quad i = 1, \ldots, N \tag{17}$$

$$x_* = \frac{1}{\sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}} \cdot (x_1, x_2, \ldots, x_N)^T \tag{18}$$

3. Identification of $N-1$ points on the constraint boundary: The line between the feasible point $\vec{x}_f$ and the infeasible point $\vec{x}_i$ cuts the constraint hyper-plane h in point h_p. We approximate h_p with binary search on this segment. The center h'_p of the last interval is an estimated point on h. Table 6 shows the number of binary search steps for the angle accuracy $\phi < 1^\circ$ and the accuracy $\phi < 0.25^\circ$ of the estimated

hyperplane h. Each experiment was repeated $100,000$ times, i.e. in each repetition a new test case with a random hyper-plane and two random points $\vec{x}_f, \vec{x}_i \in \mathbb{N}$ was generated. The error of the angle ϕ can be estimated by the number of binary search steps s as follows. Let ϕ_k be the average angle error after k binary steps and η be the accuracy improvement factor we are able to achieve with one binary search step. The following relation holds:

$$\phi_i \cdot \eta^{j-i} = \phi_j \tag{19}$$

From our experiments (see table 6) we derive an efficiency factor, i.e. an improvement of angle accuracy, of $0.53 \leq \eta \leq 0.57$. Interestingly, the number of binary search steps grows slower than the number of dimensions of the solution space.

dimension	steps ($\phi < 1°$)	mean error	steps ($\phi < 1°$)	mean error	cfe
2	9	0.85	11	0.24	3.14
5	12	0.97	15	0.16	10.50
10	14	0.68	16	0.21	21.56
20	15	0.76	17	0.25	42.96

Table 6: Number of necessary binary search steps to limit the average angle accuracy.

4. In the last step we calculate the normal vector of h using the $N-1$ points on the constraint boundary. We assume the points $h_p^{(i)}, \quad 1 \leq i \leq N-1$, represent linearly independent vectors as they have been generated by a random procedure. A successive Gram-Schmidt orthogonalization of the $(i+1)$-th vector on the i-th previously produced vectors delivers the normal vector of h.

Note that we estimate the constraint model h only one time, i.e. when the first infeasible solutions have been detected. Later update steps only concern the local support point $(h_s)_t$ of the hyper-plane h_t. We will describe the support point update step later.

3.2 Mutation Ellipsoid Rotation

Correlated mutation by Schwefel [27] rotates the axes of the hyper-ellipsoid to adapt to local properties of the fitness landscape. Experiments show that self-adaptation of the angles is to weak to automatically solve this task in many constrained problems [17]. To solve this problem we make use of the meta-model estimator for the constraint boundary. This meta-model is used for the rotation of the mutation ellipsoid. Corre-

	SA-ES	MA-ES	MM-ES (10)	MM-ES (30)
best	$1.6 \cdot 10^{-8}$	0	$2.9 \cdot 10^{-11}$	0.0
mean	$2.4 \cdot 10^{-4}$	0	$1.6 \cdot 10^{-6}$	0.0
worst	$1.6 \cdot 10^{-3}$	0	$5.6 \cdot 10^{-5}$	0.0
dev	$3.5 \cdot 10^{-4}$	$3.1 \cdot 10^{-16}$	$5.9 \cdot 10^{-6}$	0.0
ffe	22,445	927,372	18,736	11,998
cfe	39,921	1,394,023	32,960	20,183

Table 7: A comparison of correlated mutation, meta-evolution and the meta-model-based ellipsoid rotation on TR2.

lated mutations make use of $n_\alpha = \frac{N(N-1)}{2}$ additional strategy parameters, i.e. angles for the rotation of the hyper-ellipsoid. We use our model-estimator to rotate the hyper-ellipsoid by the proper angles. The n_α rotation angles can be computed knowing the normal vector $\vec{n}_h$ of h and the axes of the mutation ellipsoid. In particular, this is an easy undertaking in two dimensions.

Table 7 shows the experimental results of self-adaptive correlated mutation (SA-ES), a meta-evolutionary approach ((3,15(3,15))-MA-ES) [17], and correlated mutation using our new meta-model estimator (MM-ES) with 10 and 30 binary search steps. The self-adaptation process of the SA-ES fails to adapt the angles automatically. Obviously, the parameter space of N step sizes and n_α angles is too large to adapt by means of self-adaptation. The MA-ES is a nested ES, i.e. an outer ES evolves the angles of an inner ES that optimizes the problem itself. Of course, this approach is rather inefficient $-$ as one fitness evaluation of the outer ES causes a whole run of the inner ES $-$ but the results demonstrate that the rotation of the hyper-ellipsoid has a strong impact on the approximation capabilities on problem TR2. The MM-ES approach is capable of estimating the proper rotation angle and controlling the mutation ellipsoid to approximate the optimum. A comparison between the MM-ES approach with 10 and with 30 dimensions shows that it is advantageous to investigate more binary search steps for a better meta-model estimation. A higher accuracy of the meta-model delivers better approximation results.

4 Covariance Matrix Adaptation and the Constraint Meta-Model

Past research on constraint handling missed to concentrate on covariance matrix adaptation techniques. It is an astonishing fact that no sophisticated constraint handling techniques for these algorithms have been introduced so far. The idea of covariance matrix adaptation techniques is to adapt the distribution of the mutation operator such that the probability to reproduce steps that led to the actual population increases. This idea is similar to the estimation of distributions approaches. The covariance matrix adaptation evolution strategy (CMA-ES) was introduced by Ostermeier et al. [22, 8]. All previous CMA-ES implementations make use of death penalty as constraint handling technique. The results of the CMA-ES on problems TR and 2.40 can be found in table 8. Amazingly, the CMA-ES is able to cope with the low success rates around the optimum of the TR-problems. On problem TR2 the average number of infeasible solution during the approximation is 44%. This indicates that a reasonable adaptation of the mutation ellipsoid takes place $-$ otherwise we would expect a rate of $\approx 50\%$ infeasible solutions and a decrease of steps sizes. An analysis of the angle between the main axis of the mutation ellipsoid and the constraint function converges to zero, the same do the step sizes during approximation of the optimum. On problem 2.40 about 64% of the produced solutions are infeasible.

CMA-ES (DP)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.8 \cdot 10^{-16}$	6,754	12,019
2.40	0.0	0.0	$1.3 \cdot 10^{-13}$	19,019	71,241

Table 8: Experimental analysis of the CMA-ES with death penalty.

4.1 Checking Feasibility

Although the CMA-ES is able to find the optimum of both problems, we try to improve the search with the constraint meta-model. In our first approach we use the model to check the feasibility of mutations during the mutation procedure in order to reduce constraint function evaluations. Nevertheless, potentially feasible solutions are checked for feasibility with a real constraint function evaluation. For this purpose an exact estimation of the constraint boundary in necessary. Two errors for the feasibility prediction of individual $\vec{x}_t$ are possible:

1. The model predicts $\vec{x}_t$ is feasible, but it is not. Points of this category are examined for feasibility. This will cause an unnecessary constraint function evaluation.

2. The model predicts $\vec{x}_t$ is infeasible, but it is feasible. The individual will be discarded, but may be a very good approximation of the optimum.

We introduce a *safety margin* s, i.e. a shift of the estimated constraint boundary into the infeasible direction. This safety margin ensures that errors of type 2 are reduced. We set s to the distance d of the mutation ellipsoid center c_m and the estimated constraint boundary h. Hence, the distance between c_m and the h becomes $2d$. As mentioned in section 3, a regular update of the constraint boundary support point h_s is necessary. Again, we have to distinguish between two conditions in each iteration. Let d_{t_0} be the distance between the mutation ellipsoid center $(c_m)_{t_0}$ and the constraint boundary h_{t_0} at time t_0 and let k be the number of binary search steps to achieve the angle accuracy of $\delta < 0.25°$.

1. The search $(c_m)_t$ approaches h_t: If distance d_t between h_t and $(c_m)_{t_0}$ becomes smaller than $d_{t_0}/2^k$, a re-estimation of the support point h_s is reasonable.

2. The search $(c_m)_t$ moves parallel to h_t: An exceeding of distance

$$(c_m)_{t_0} - (c_m)_t = \sqrt{\frac{1}{\tan(\phi)^2} + 4} \cdot d_{t_0} \tag{20}$$

   with $\phi = 0.25 \cdot (0.57)^{3k}$ causes a re-estimation of h_t.

We use $4k$ binary steps on the line between the current infeasible solutions and $(c_m)_t$ to find the new support point $(h_s)_t$. Table 9 shows the results of the CMA-ES with feasibility check using the constraint meta-model. We can observe a significant saving of fitness and constraint evaluations with a high approximation capability.

CMA-ES (Check)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$6.9 \cdot 10^{-16}$	6,780	7,781
2.40	0.0	0.0	$1.8 \cdot 10^{-13}$	19,386	34,254

Table 9: Results of the CMA-ES with meta-model feasibility checking.

4.2 Repairing Infeasible Solutions

The repair approach projects infeasible mutations onto the constraint boundary h_t. We assume the angle error ϕ that can again be estimated by the number of binary search steps s, see equation 19. In our approach we elongate the projection vector by length s. Figure 1 illustrates the situation. Let $(h_s)_t$ be the support point of the hyper-plane and let x_i be the infeasible solution. It holds $a^2 + b^2 = c^2$ and $l/b = \tan\phi$. We get

$$s = \sqrt{a^2 - c^2} \cdot \tan\phi. \tag{21}$$

The elongation of the projection into the potentially feasible region guarantees

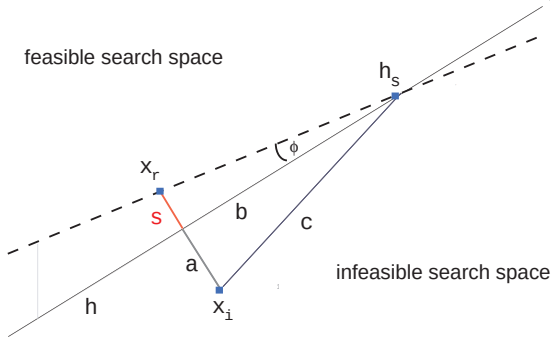

Figure 1: The elongation of the projection of infeasible solution x_i onto the constraint boundary h by length s results in a repaired feasible point x_r.

feasibility of the repaired individuals. Nevertheless, it might prevent fast convergence, in particular in regions far away from the hyper-plane support point h_s as s grows with increasing length of c. In our approach we update the center of the hyper-plane for an update of accuracy every $t = 10$ generations. The results of the CMA-ES repair algorithm can be found in table 10. We observe a significant decrease of fitness function evaluations, in particular on problem TR2. The search concentrates on the boundary of the infeasible search space, in particular on the feasible site. Of course, no saving of constraint function evaluations could have been expected.

CMA-ES (Repair)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.5 \cdot 10^{-16}$	3,432	5,326
2.40	0.0	0.0	$9.1 \cdot 10^{-14}$	16,067	75,705

Table 10: Results of the CMA-ES with repair mechanism.

5 Summary and Outlook

Continuous constraint handling frequently suffers from low success rates at the constraint boundary and makes optimization in the vicinity of the infeasible solution space notedly difficult. Among the various constraint handling methods, the use of meta-models turns out to be the most promising approach. The better the constraints are known, the more information can be investigated into the search process. We plan to extend the constraint meta-modeling approach in many kinds of ways. At first, all the proposed approaches will be tested on further practical and artificial constrained test problems. Latest experiments show that a combination of the proposed approaches, i.e. repairing and feasibility checking at the same time is a promising undertaking. In practice, constraints are frequently far away from being linear. Hence, we plan to extend the meta-modeling to nonlinear constraints.

References

[1] J. C. Bean and A. B. Hadj-Alouane. A dual genetic algorithm for bounded integer programs. Technical report, University of Michigan, 1992.

[2] S. V. Belur. CORE: Constrained optimization by random evolution. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1997 Conference*, pages 280–286, Stanford University, California, July 1997. Stanford Bookstore.

[3] H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[4] C. A. Coello Coello. Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[5] C. A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[6] A. Fiacco and G. McCormick. The sequential unconstrained minimization technique for nonlinear programming - a primal-dual method. *Mgmt. Sci.*, 10:360–366, 1964.

[7] C. Floudas and P. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Springer, Berlin and Heidelberg, 1990.

[8] N. Hansen. The cma evolution strategy: A tutorial. Technical report, TU Berlin, ETH Zürich, 2005.

[9] F. Hoffmeister and J. Sprave. Problem-independent handling of constraints by use of metric penalty functions. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the 5th Conference on Evolutionary Programming - EP 1996*, pages 289–294, Cambridge, February 1996. MIT Press.

[10] A. Homaifar, S. H. Y. Lai, and X. Qi. Constrained Optimization via Genetic Algorithms. *Simulation*, 62(4):242–254, 1994.

[11] F. Jimenez and J. L. Verdegay. Evolutionary techniques for constrained optimization problems. In H.-J. Zimmermann, editor, *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT 1999)*, Aachen, 1999. Verlag Mainz.

[12] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In D. B. Fogel, editor, *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[13] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[14] O. Kramer. Premature convergence in constrained continuous search spaces. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, pages 62–71, Berlin, Heidelberg, 2008. Springer-Verlag.

[15] O. Kramer. *Self-Adaptive Heuristics for Evolutionary Computation*. Springer, Berlin, 2008.

[16] O. Kramer, S. Brügger, and D. Lazovic. Sex and death: towards biologically inspired heuristics for constraint handling. In *Proceedings of the 9th conference on genetic and evolutionary computation - GECCO 2007*, pages 666–673, New York, 2007. ACM Press.

[17] O. Kramer and H.-P. Schwefel. On three new approaches to handle constraints within evolution strategies. *Natural Computing*, 5(4):363–385, 2006.

[18] O. Kramer, C.-K. Ting, and H. K. Büning. A new mutation operator for evolution strategies for constrained problems. In *Proceedings of the IEEE Congress on Evolutionary Computation - CEC 2005*, pages 2600–2606, 2005.

[19] A. Kuri-Morales and C. V. Quezada. A universal eclectic genetic algorithm for constrained optimization. In *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT 1998*, pages 518–522, Aachen, September 1998. Verlag Mainz.

[20] G. Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43:645–646, 1972.

[21] Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[22] A. Ostermeier, A. Gawelczyk, and N. Hansen. A derandomized approach to self adaptation of evolution strategies. *Evolutionary Computation*, 2(4):369–380, 1994.

[23] J. Paredis. Co-evolutionary constraint satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature - PPSN 1994*, pages 46–55, Berlin, 1994. Springer.

[24] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control 1994*, pages 97–102, Plymouth, 1994. University of Plymouth.

[25] M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV*, pages 245–254, Berlin, September 1996. Springer.

[26] M. Schoenauer and S. Xanthakis. Constrained GA optimization. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms - ICGA 1993*, pages 573–580, San Francisco, July 1993. Morgan Kaufman Publishers Inc.

[27] H.-P. Schwefel. Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik, TU Berlin, July 1974.