Check for updates

Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 185063, 11 pages
doi:10.1155/2010/185063

Review Article

A Review of Constraint-Handling Techniques for Evolution Strategies

Oliver Kramer

International Computer Science Institute, Berkeley, CA 94704, USA

Correspondence should be addressed to Oliver Kramer, okramer@icsi.berkeley.edu

Received 24 September 2009; Accepted 6 January 2010

Academic Editor: Chuan-Kang Ting

Copyright © 2010 Oliver Kramer. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Evolution strategies are successful global optimization methods. In many practical numerical problems constraints are not explicitly given. Evolution strategies have to incorporate techniques to optimize in restricted solution spaces. Famous constraint-handling techniques are penalty and multiobjective approaches. Past work has shown that in particular an ill-conditioned alignment between the coordinate system of Gaussian mutation and the constraint boundaries leads to premature convergence. Covariance matrix adaptation evolution strategies offer a solution to this alignment problem. Last, metamodeling of the constraint boundary leads to significant savings of constraint function calls and to a speedup by repairing infeasible solutions. This work gives a brief overview over constraint-handling methods for evolution strategies by demonstrating the approaches experimentally on two exemplary constrained problems.

1. Introduction

Many continuous optimization problems in practical applications are subject to constraints [1]. 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 evolutionary computation. This paper is devoted to constraint-handling techniques that have been developed, in particular for evolution strategies. It summarizes our line of research of the last years in the field of constraint-handling and premature step-size reduction [2–7]. In real-valued solution spaces a constrained problem can be hard to solve due to a coordinate system alignment problem that leads to premature fitness stagnation. We review not only various general approaches like penalty functions, but also specialized approaches that have been developed to solve coordinate alignment problems, by summarizing each constraint-handling method, stating experimental results on two exemplary test functions and discussing advantages and disadvantages.

The remainder of this section gives a brief introduction to evolution strategies, constrained problems, and a taxonomy of constraint-handling techniques. Section 2 introduces three examples from the famous family of penalty functions. A bioinspired multiobjective approach is reviewed in Section 3. The methods that concentrate on coordinate system alignment are presented in Section 4, while Section 5 is devoted to metamodeling of the constraint boundary.

1.1. Evolution Strategies. Evolution strategies (ES) are a family of strong stochastic methods for global optimization. Developed by Rechenberg [8] and Schwefel [9], they have become famous for global numerical optimization, that is, nonconvex optimization in $\mathbb{R}^N$. In each iteration λ offspring solutions are produced and the μ best are selected as parents for the following generation. An important basis of ES is the self-adaptive Gaussian mutation operator that we briefly repeat in this context. An individual $\mathbf{a}$ of a (μ^+,λ)-ES with the N-dimensional objective variable vector $\mathbf{x} \in \mathbb{R}^N$ is mutated in the following way:

$$\begin{aligned} \mathbf{x}' &:= \mathbf{x} + \mathbf{z}, \\ \mathbf{z} &:= (\sigma_1 \mathcal{N}_1(0,1), \ldots, \sigma_N \mathcal{N}_N(0,1)), \end{aligned} \tag{1}$$

while $\mathcal{N}_i(0,1)$ delivers a Gaussian distributed number. The strategy parameter vector undergoes mutation—a typical variation of σ—with log-normal mutation:

$$\sigma' := e^{(\tau_0 \mathcal{N}_0(0,1))} \cdot \left(\sigma_1 e^{(\tau_1 \mathcal{N}_1(0,1)}, \ldots, \sigma_N e^{(\tau_1 \mathcal{N}_N(0,1)}\right), \quad (2)$$

as crossover operator arithmetic recombination is applied in most cases. For a detailed introduction to ES we recommend the introduction by Beyer and Schwefel [10] or the introductory chapter to ES in Eiben's book [11].

1.2. Constrained Problems. 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. 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 $\mathbb{R}^N$ that minimizes the objective function $f(\mathbf{x})$, in symbols as:

$$\begin{aligned} f(\mathbf{x}) \longrightarrow \min!, \quad & \mathbf{x} \in \mathbb{R}^N \quad \text{subject to} \\ \text{inequalities} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, n_1, \\ \text{equalities} \quad & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, n_2\,. \end{aligned} \quad (3)$$

A feasible solution $\mathbf{x} \in \mathbb{R}^N$ satisfies all n_1 inequality and n_2 equality constraints. A feasible solution that minimizes $f(\cdot)$ is termed as 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. We assume that the evaluations of the constraint functions are computationally expensive and that the return values are boolean and provide the information of whether the solution is feasible or not. In order to be able to develop more advanced constraint-handling techniques, for example, repair or feasibility check approaches, metamodels of the constraint function can be built with certain assumptions, that is, to be linear, quadratic, and so forth.

The two following test functions excellently demonstrate the phenomenon of premature fitness stagnation that will be discussed in the following sections and that is a challenge for most constraint-handling techniques. The two functions will be used for the discussion of the methods reviewed in the current paper. Problem 2.40—taken from Schwefel's artificial test problems [10]— exhibits a linear objective function and an optimum with five active linear constraints. The problem is to minimize

$$f_{2.40}(\mathbf{x}) = -\sum_{i=1}^{5} x_i, \quad (4)$$

subject to

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

with minimum $\mathbf{x}^* = (5000, 0, 0, 0, 0)^T$ and $f(\mathbf{x}^*) = -5000$.

The second problem is called tangent problem (TR). It is based on the sphere model subject to one linear constraint:

$$f_{\mathrm{TR}}(\mathbf{x}) = \sum_{i=1}^{N} x_i \quad \text{with } g_{\mathrm{TR}}(\mathbf{x}) = \sum x_i - N > 0 \quad (6)$$

with $\mathbf{x}^* = (1, \ldots, 1)^T$ and $f(\mathbf{x}^*) = N$. The success rates on TR get worse when approximating the optimum. In this paper we will focus on the TR problem with $N = 2$ dimensions, denoted as TR2.

1.3. A Brief Taxonomy of 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.

(i) *Penalty functions* decrease the fitness of infeasible solutions by taking the number of infeasible constraints or the distance to feasibility into account [12–16]. The history of penalty functions began with the sequential unconstrained minimization technique by Fiacco and McCormick [13] in which the constrained problem is solved by a sequence of unconstrained optimizations. The penalty factors are stepwise intensified. In similar approaches penalty factors can be defined statically [14] or depending on the number of satisfied constraints [16]. They can dynamically depend on the number of generations as Joines and Houck propose [15]:

$$\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + (C \cdot t)^{\alpha} \cdot G(\mathbf{x}), \quad (7)$$

at generation t, parameters C and α are user defined; typical settings are $C = 0.5$, $\alpha = 1$, or 2. $G(\mathbf{x})$ is a measure for the constraint violation. A frequent definition is $G(\mathbf{x}) = \sum_{i=1}^{n_1} \max[0, g_i(\mathbf{x})]^{\beta} + \sum_{j=1}^{n_2} |h_j(\mathbf{x})|^{\gamma}$ with factors $\beta \geq 1$ and $\gamma \geq 1$. Penalties can be adapted according to an external cooling scheme [15] or by adaptive heuristics [12]. In the death penalty approach [5] infeasible solutions are rejected and new solutions are created until enough feasible ones exist. In the segregated genetic algorithm by Riche et al. [17] two penalty functions, a weak one and an intense one, are calculated in order to surround the optimum. In the coevolutionary penalty function approach by Coello Coello [18] the penalty factors of an inner evolutionary algorithm are adapted by an outer evolutionary algorithm. Some methods are based on the assumption that any feasible solution is better than any infeasible one [19, 20]. Examples are the metric penalty functions by Hoffmeister and Sprave [21]. Feasible solutions are compared using the objective function while infeasible solutions are compared considering the satisfaction of constraints. Similar is the approach by

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Oyman et al. [22]. Their fitness function depends on the parent and children population at every generation and, therefore, becomes a dynamic approach.
In his approach called stochastic-ranking Runarsson [23] he uses metamodels to predict both fitness functions values and penalties based on constraint violations. From this point of view the approach is related to methods that are based on metamodeling the constraint boundary.

(ii) *Repair algorithms* either replace infeasible solutions or only use the repaired solutions for evaluation of their infeasible pendants [24, 25]. This class of algorithms can also be seen as local search methods that reduce the constraint violation. The repair algorithm generates a feasible solution from an infeasible one. In the Baldwinian case, the fitness of the repaired solution replaces the fitness of the original solution. In the Lamarckian case, the feasible solution overwrites the infeasible one. In general, defining a repair algorithm can be as complex as solving the problem itself.

(iii) *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 [25–27]. They map a genotype into a feasible phenotype. By this means even quite different genotypes may be mapped onto the same phenotype. Eiben and Smith [11] define decoders as a class of mappings from the genotype space $\mathcal{S}'$ to the feasible regions $\mathcal{F}$ of the solution space $\mathcal{S}$ with the following properties: every $z \in \mathcal{S}'$ must map to a single solution $s \in \mathcal{F}$, every solution $s \in \mathcal{F}$ must have at least one representation $s' \in \mathcal{S}'$, and every $s \in \mathcal{F}$ must have the same number of representations in $\mathcal{S}'$ (this need not be one).

(iv) *Feasibility preserving representations and operators* force candidate solutions to be feasible [28, 29]. A famous example is the GENOCOP algorithm [27] that reduces the problem to convex search spaces and linear constraints. A predator-prey approach to handle constraints is proposed by Paredis [28] using two separate populations. Schoenauer and Michalewicz [29] propose special operators that are designed to search regions in the vicinity of active constraints. A comprehensive overview to decoder-based constraint-handling techniques is given by Coello [25] and also by Michalewicz and Fogel [27].

(v) *Multiobjective optimization* techniques are based on the idea of handling each constraint as an objective [30–35]. Under this assumption many multiobjective optimization methods can be applied. Such approaches were used by Parmee and Purchase [34], Jimenez and Verdegay [32], Coello Coello [31], and Surry et al. [36]. In the behavioral memory-method by Schoenauer and Xanthakis [35] the EA concentrates on minimizing the constraint violation of each constraint in a certain order and optimizing the objective function in the last step.

Of course, constraint-handling methods exist that do not fit into the taxonomy. Montes and Coello Coello [37] introduced a technique based on a multimembered ES with a feasibility comparison mechanism. The ϵ-constrained differential evolution approach by Takahama and Sakai [38] combines the usage of an ϵ for equality constraints with differential evolution. The dynamic multiswarm particle optimizer by Liang and Suganthan [39] makes use of a set of subswarms concentrating on different constraints. It is combined with sequential quadratic programming as a local search method. The approach of Mezura-Montes et al. [40] combines differential evolution, different mutation operators to increase the probability of producing better offspring, three selection criteria, and a diversity mechanism. Mezura-Montes [41] approach gives a survey of constraint-handling methods for evolutionary algorithms.

In the following section we will compare various approaches from different fields and compare them, in particular with regard to the mentioned premature step-size problem. The next section shows this problem experimentally.

Table 1: Experimental results of the death penalty method.

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

2. Penalty Methods

Evolutionary search is guided by the quality of its candidate solutions. Consequently, an obvious solution to constraint-handling is to deteriorate the fitness of infeasible methods [11, 25]. Here we review three penalty functions exemplarily. Death penalty is the simplest way, but wastes comparably many constraint function calls. Paragraph 2.2 is a typical penalty technique where the solutions are penalized with regard to the progress of the search. The death penalty step control approach that prevents premature step-size reduction is reviewed in Section 2.3.

2.1. Death Penalty. First of all, we will analyze the behavior of death penalty, that is, simply discarding infeasible offspring solutions [42, 43]. This is the first time we can observe premature fitness stagnation. Table 1 shows the corresponding results of a (15,100)-ES with the following settings on problems 2.40 and TR2. We use the mutation introduced in Section 1.1 with settings $\tau_0 = (\sqrt{2N})^{-1}$ and $\tau_1 = (\sqrt{2\sqrt{N}})^{-1}$ and arithmetic recombination with $\rho = 2$ randomly chosen parents. 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, and dev describe the achieved fitness (difference between the optimal fitness and the fitness of the best solution $|f(\mathbf{x}^*) - f(\mathbf{x}^{\text{best}})|$) 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.

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 2: Experimental results of the dynamic penalty function by Joines and Houck [15] on problems TR2 and 2.40.

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

We can summarize the behavior of death penalty mentioning the advantage that *death penalty* is easy to implement. The disadvantages are that *death penalty* is inefficient as many infeasible tries are wasted, and it suffers from premature convergence. The following methods aim at preventing premature convergence.

2.2. Dynamic Penalty Functions. 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 [15] that is based on adding a penalty on infeasible solutions

$$\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + (C \cdot t)^{\alpha} \cdot \sum_{i=1}^{n_1} G_i^{\beta} \tag{8}$$

with parameters C, α, β and the constraint violation $G_i(\mathbf{x}) = \sum_{i=1}^{n_1} \max[0, g_i(\mathbf{x})]^{\beta}$. 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 whether caused by discarding or penalizing. Consequently, we can summarize as follows: *dynamic penalty functions* are easy to implement, and no feasible starting point is required. But the disadvantages are that *dynamic penalty functions* suffer from premature convergence. Related work on penalty functions can be found in [12–16].

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 for the mutation strengths σ_i with $1 \leq i \leq N$:

$$\sigma_i \geq \epsilon. \tag{9}$$

This is exactly what the death penalty step control evolution strategy (DSES) is aiming at [5]. 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: in every ϖ infeasible trial, ϵ is reduced by a factor $0 < \vartheta < 1$ according to the equation

$$\epsilon' := \epsilon \cdot \vartheta. \tag{10}$$

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*, that is, 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 following results: *death penalty step control* is easy to implement, and shows an improvement of the approximation of optima with active constraints. But the disadvantages are that *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. A more detailed experimental analysis of the DSES can be found in [4, 5].

3. A Multiobjective Bioinspired Approach

A familiar variant to handle constraints is to treat each constraint—or an aggregated sum of all constraints—and the objective function as separate objectives in a multiobjective formulation. Similar approaches have been introduced in the past [30–35]. Here we review a similar constraint-handling technique that treats the fulfillment of constraints and the optimization of the objective function as separate objectives that are optimized using a population specific selection scheme. The bioinspired concept offers an answer to the problem of low success rates: our two-sex evolution strategy (Kramer and Schwefel [5]) 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, that is, a two-step selection operator for individuals of sex *c* similar to the operator by Hoffmeister and Sprave [21]. For a list of TSES variants and modifications we refer to [5]. The populations

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 3: Experimental results of the death penalty step control evolution strategy.

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 4: Experimental results of the two-sex evolution strategy on TR2 and 2.40.

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

are noted as $(\mu_o + \mu_c, \lambda_o + \lambda_c)$—the index determines the sex, that is, o for objective function and c for constraints.

Table 4 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 [4]—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-sex evolution strategy* improves the approximation of optima with active constraints, allows infeasible starting points, saves constraint function evaluations, for example, in comparison to the DSES, and is quite robust to parameter changes. But the disadvantages are that the *two-sex evolution strategy* still consumes many fitness function evaluations; on some problems it may still suffer from low success rates, for example, on TR2.

4. Coordinate Alignment Techniques

In real-valued optimization the coordinate system plays an important role. If the coordinate system of the mutation operators, for example, of Gaussian mutation, is not aligned to the coordinate system of the objective function—and this is frequently the case in black-box optimization—undesirable effects may occur like premature step-size reduction.

4.1. Premature Step-Size Reduction. The phenomenon of premature step-size reduction at the constraint boundary has been analyzed in [2]—in particular for the condition that the optimum lies on the constraint boundary or even in a vertex of the feasible search space. In such cases the evolutionary algorithm frequently suffers from low success probabilities near the constraint boundaries. Under simple conditions, that is, 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 proven. Figure 1 illustrates the reason for premature step size reduction. We assume the simplified case in which mutations are produced on the boundary of the circles.

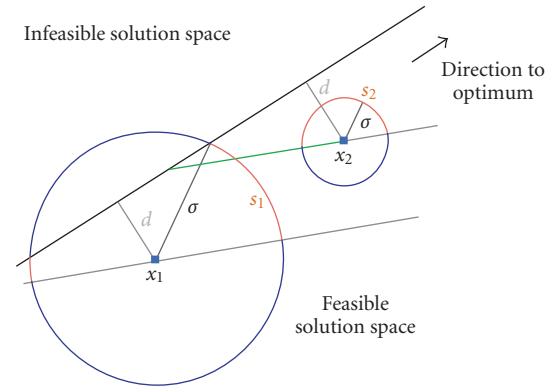

Figure 1: Illustration of the success probabilities at the constraint boundary. In this simplified model we assume that mutations are produced uniformly on the boundary of the circles. Both solutions x_1 and x_2 lie close to the constraint boundary with distance d. In case of small step sizes $\sigma < d$, the success probability $(s_i/(2\pi\sigma))$ is higher than in the case of $\sigma > d$.

in case of large mutation strengths (x_1) with $\sigma > d$ the region of success, that is, the marked part s_1 of the circles, is smaller in comparison to the whole circle than in the case that the circle is not cut by the constraint boundary (x_2). Consequently, the probability to produce successful mutations is higher for small step sizes and these mutations are favored during optimization. This is a coordinate system alignment problem: In case of N independent step sizes and coordinate rotation the mutation circle can adapt to a mutation ellipsoid whose region of success is not restricted by the constraint boundary.

Arnold and Brauer [44] analyzed the behavior at the boundary of linear constraints and models the distance between the search point and the constraint plane with a Markov chain. Furthermore, they discuss the working of step length adaptation mechanisms based on success probabilities.

4.2. 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 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 [7]. A self-adaptive bias coefficient vector ξ 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.

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

The mutation operator adapts the bias direction within the interval -1 (for left) and 1 (for right) in each of the N dimensions:

$$\xi = (\xi_1, \ldots, \xi_N) \quad \text{with} -1 \leq \xi_i \leq 1. \tag{11}$$

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 $\mathbf{b} = (b_1, \ldots, b_N)$ is defined by

$$b_i = \xi_i \cdot \sigma_i. \tag{12}$$

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:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} + (\sigma_1 \mathcal{N}_1(0,1) + b_1, \ldots, \sigma_N \mathcal{N}_N(0,1) + b_N) \\ &= \mathbf{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)). \end{aligned} \tag{13}$$

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

$$\xi_i' = \xi_i + \gamma \cdot \mathcal{N}(0,1), \quad i = 1, \ldots, N, \tag{14}$$

with parameter γ determining the mutation strength on the bias. The biased mutation operator (BMO) biases the mean of mutation and enables the ES 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 σ. 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 χ_t and χ_{t+1} of successive generations. Let χ_t be the center of the population at generation t:

$$\chi_t = \sum_{i=1}^{\mu} \mathbf{x}_i. \tag{15}$$

The normalized descent direction ξ of two successive population centers χ_t and χ_{t+1} is

$$\xi = \frac{\chi_{t+1} - \chi_t}{\left| \chi_{t+1} - \chi_t \right|}. \tag{16}$$

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

$$\mathbf{x}' = \mathbf{x} + (\sigma_1 \mathcal{N}_1(\xi_1, 1), \ldots, \sigma_N \mathcal{N}_N(\xi_N, 1)). \tag{17}$$

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, that is, 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 5. 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.

Table 5: Experimental results of the biased mutation variants BMO and DMO.

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

We can conclude that *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 that *biased mutation* consumes many fitness and constraint function evaluations, and on some problems it may still suffer from low success rates.

4.3. Mutation Ellipsoid Rotation. Correlated mutation by Schwefel [45] rotates the axes of the hyperellipsoid to adapt to local properties of the fitness landscape. Three ways are possible to rotate the mutation ellipsoid with the help of $N_\alpha = N(N-1)/2$ possible rotation angles:

(1) a self-adaptive rotation—in this case the N_α rotation angles become strategy parameters and the algorithm has to tune itself,

(2) a rotation with the help of a coevolutionary approach,

(3) with a metamodel of the constraint boundary that delivers the orientation of the constraint boundary.

Table 6 shows the experimental results of self-adaptive correlated mutation (SA-ES), a metaevolutionary approach ((3,15(3,15))-MA-ES) [5], and correlated mutation using the metamodel estimator (MM-ES) with 10 and 30 binary search steps. Correlated mutations make use of N_α additional strategy parameters, that is, angles for the rotation of the hyperellipsoid. The self-adaptation process of the SA-ES fails to adapt the angles automatically. The parameter space of N step sizes and N_α angles is too large to adapt successfully by means of self-adaptation. The MA-ES is a nested ES, that is, 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 on the original problem—but the results demonstrate that the rotation of the hyperellipsoid has a strong impact on

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Table 6: A comparison of correlated mutation, metaevolution, and the metamodel-based ellipsoid rotation on TR2.

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

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

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. We use the linear metamodel that will be introduced in Section 5. The N_α rotation angles can be computed estimating the normal vector $\mathbf{n}_h$ of the estimated hyperplane h and the axes of the mutation ellipsoid. This is an easy undertaking in two dimensions. A comparison between the MM-ES approach with 10 and with 30 binary search steps shows that it is advantageous to invest search for a precise metamodel estimation: a higher accuracy of the metamodel delivers better approximation results.

Obviously, the coordinate system alignment problem is solved with the mutation ellipsoid rotation. But the self-adaptive rotation does not lead to satisfying results, while the metaevolutionary approach is inefficient. In the following paragraph we will investigate whether the covariance matrix adaptation techniques, which are designed to align coordinate systems, are able to adapt their covariance matrix to constrained problems automatically without a metamodel.

4.4. Covariance Matrix Techniques. 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. Nevertheless, we will now analyze whether the coordinate system alignment problem can be solved with covariance matrix adaptation using death penalty. 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 Hansen [46] and Ostermeier [47]. The results of the CMA-ES on problems TR2 and 2.40 can be found in Table 7. Amazingly, the CMA-ES is able to cope with the low success rates around the optimum of the TR problem. We observed that the average number of infeasible solutions during the approximation is 44%. This indicates that a reasonable adaptation of the mutation ellipsoid takes place. An analysis of the angle between the main axis of the mutation ellipsoid and the constraint function shows that it converges to zero, the same do the step sizes during approximation of the optimum. Hence, the coordinate system alignment is successful.

We can conclude that the CMA-ES is able to align the coordinate system automatically without a metamodel. Recent results have shown that an acceleration can be achieved if the covariance matrix is rotated with the help of a metamodel exactly at the time when the constraint boundary is reached [3].

5. Metamodeling of Constraints

In black-box scenarios the constraint boundaries are not explicitly given. Metamodeling of constraints allows advanced constraint-handling methods. Metamodels can be used for various purposes, for example, for checking the feasibility and for repair of infeasible mutations, and—like we have seen in the previous section—for control of mutation ellipsoids and covariance matrices. Metamodeling of objective functions has developed to a successful standard in evolutionary optimization [48–50].

5.1. Linear Constraint Estimation. For constraint metamodeling various classification and regression methods can be applied. For the case of linear constraints a metamodel that is based on sampling N infeasible points and binary search on the segments to the last feasible point has been developed [3]. The approach works as follows: first, the center point of the model estimator is determined. When the first infeasible offspring individual q_1 is produced, the feasible parent $\mathbf{x}_f$ is the center of the corresponding metamodel estimator and the distance becomes radius r of the model estimator. Then, random points are generated on the surface of a hypersphere. Point $\mathbf{x}_f$ is the center of a hypersphere with radius r, such that the constraint boundary is cut. In N dimensions $N-1$ additional infeasible points q_i, $1 \leq i \leq N-1$ have to be produced. The model estimator produces the infeasible points by sampling on the surface of a hypersphere with radius r until a sufficient number of infeasible points are produced. The points on the surface are sampled randomly with uniform distribution using the method of Marsaglia [51]. In the first step the algorithm produces $N-1$ Gaussian distributed points and scales the numbers to length 1. Further scaling and shifting yields N randomly distributed point on the hypersphere surface.

In a next step the binary search procedure is applied to identify N points $s_1, \ldots, s_N$ *on* the constraint boundary: the line between the feasible point $\mathbf{x}_f$ and the ith infeasible point q_i cuts the real constraint hyperplane h^* in point s_i^*. We approximate s_i^* with binary search on this segment. The center s_i of the last interval defined by the last points of the binary search is an estimation of point s_i^* on h_0. Figure 2 illustrates the situation. With regard to the estimated angle error ϕ, the real hyperplane lies between h_1^* and h_2^*.

In the last step we calculate the normal vector $\mathbf{n}_0$ of h_0 using the N points on the constraint boundary. We assume that the points s_i, $1 \leq i \leq N$, represent linearly independent vectors as the endpoints of the lines they lie on

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

have been generated in a random procedure. A successive Gram-Schmidt orthogonalization of the $(i + 1)$th vector on the ith previously produced vectors delivers the normal vector $\mathbf{n}_0$ of h_0. Note that we estimate the normal vector $\mathbf{n}_0$ of the linear constraint model h_0 only one time, that is, when the first infeasible solutions have been detected. Later update steps only concern the local support point p_t of the hyperplane (hence, in iteration t the linear model h_t is specified by normal vector $\mathbf{n}_0$ and current support point p_t). At the beginning, any of the points s_i may be the support point p_0. For later update steps two cases have to be distinguished. Let d_{t_0} be the distance between the mutation ellipsoid center c_{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 (i.e., the center of the mutation ellipsoid) c_t approaches h_t: if distance d_t between h_t and c_{t_0} becomes smaller than $d_{t_0}/2^k$, a reestimation of the support point p_t is reasonable.

(2) The search c_t moves parallel to h_t: an exceeding of distance

$$c_{t_0} - c_t = \sqrt{\frac{1}{\tan(\phi)^2} + 4} \cdot d_{t_0} \tag{18}$$

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

We use $4k$ binary steps on the line between the current infeasible solutions and c_t to find the new support point p_t.

For nonlinear constraints other regression or classification techniques may be taken into account like support vector regression or support vector machines [52].

5.2. Feasibility Check. A metamodel can be used to check the feasibility of new solutions in order to reduce constraint function evaluations [3]. For this purpose an exact estimation of the constraint boundary is necessary. Potentially feasible solutions are checked for feasibility with a real evaluation of the constraint function. Two errors for the feasibility prediction of individual $\mathbf{x}_t$ are possible.

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

Exemplarily, we take the linear constraint metamodel of the previous paragraph into account and test the feasibility check approach. A *safety margin* δ can reduce the number of errors of type 2. We set δ to the distance d of the mutation ellipsoid center c and the estimated constraint boundary h_t. Hence, the distance between c and the shifted constraint boundary h'_t becomes $2d$. A regular update of the constraint boundary support point p_t is necessary; see previous Section 5.1. Table 8 shows the results of the CMA-ES with feasibility check using the constraint metamodel.

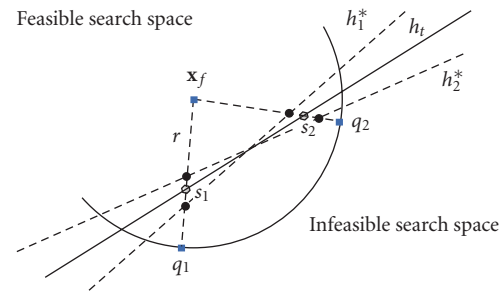

Figure 2: Procedure to estimate the constraint boundary h_0 in two dimensions: the method performs binary search on the segments between a feasible point $\mathbf{x}_f$ and each two infeasible points q_1, q_2 to estimate two points s_1, s_2 on the metamodel.

Table 8: Results of the CMA-ES with feasibility check based on the linear metamodel.

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 repair mechanism based on the linear metamodel.

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

We can observe a significant saving of fitness and constraint evaluations with a high approximation capability.

5.3. Solution Repair. The repair approach projects infeasible mutations onto the constraint boundary h_t. We assume the angle error ϕ that can be estimated by the number of binary search steps k. In the solution repair approach the projection vector is elongated by length δ. Figure 3 illustrates the situation. Let p_t be the support point of the hyperplane h_t at time t and let x_i be the infeasible solution. It holds that $a^2 + b^2 = d^2$ and $\delta/b = \tan\phi$. We get

$$\delta = \sqrt{a^2 - d^2} \cdot \tan\phi. \tag{19}$$

The elongation of the projection into the potentially feasible region guarantees feasibility of the repaired individuals. Nevertheless, it might prevent fast convergence, in particular in regions far away from the hyperplane support point p_t as δ grows with increasing length of d. The center of the hyperplane is updated every 10 generations. The results of the CMA-ES repair algorithm can be found in Table 9. 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.

6. Summary

Many constraint-handling methods exist for evolution strategies, at the head penalty functions. Due to low success

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

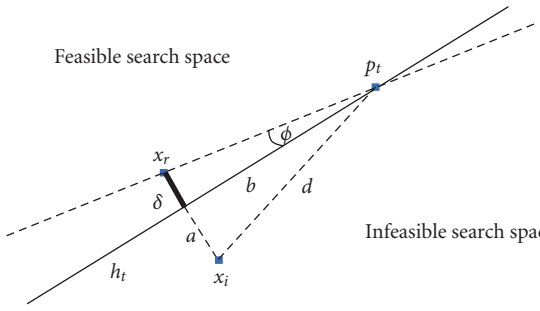

Figure 3: The elongation of the projection of infeasible solution x_i onto the estimated constraint boundary h_t by length δ ensures that the repaired point x_r is feasible.

Table 10: Results of the CMA-ES with covariance matrix rotation, feasibility check, and repair mechanism [3].

CMA-ES (all)	best	mean	dev	ffe	cfe
TR2	0.0	0.0	$5.1 \cdot 10^{-16}$	3,249	3,650
2.40	0.0	0.0	$9.1 \cdot 10^{-14}$	11,216	30,069

rates at the constraint boundary, ES without coordinate alignment techniques often fail to find the optima in the vertex of the feasible solution space. The death penalty step control approach and the multiobjective biologically inspired two-sex ES prevent a premature step-size reduction on some problems, but its success depends on proper parameter settings. Low success rates at the constraint boundary can be increased with coordinate system alignment techniques. A first step into this direction is biased mutation techniques, that is, biased mutation and descent biased mutation. Much better results can be achieved with metamodel-based mutation ellipsoid rotation. This rotation cannot be achieved self-adaptively, but automatically with covariance matrix adaptation mechanisms. The latter shows excellent results, even on hard problems like TR2. Further improvements of the CMA-ES can be achieved with metamodeling: constraint boundary surrogates can be used for prediction of feasibility of mutations and for repair of infeasible solutions. At last, Table 10 summarizes the best results that could be achieved on the two problems combining the CMA-ES with covariance matrix rotation, feasibility check, and repair of infeasible solutions.

Metamodeling of constraints will probably become more and more important for future research. Nonlinear models will increase the accuracy of feasibility prediction. Advanced regression methods will improve the accuracy of repaired infeasible solutions. Further constraint-handling methods are imaginable like adaptation of mutation probability distributions and covariance matrices—also with non-linear metamodels.

References

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

[2] O. Kramer, "Premature convergence in constrained continuous search spaces," in *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, pp. 62–71, Springer, Dortmund, Germany, 2008.

[3] O. Kramer, A. Barthelmes, and G. Rudolph, "Surrogate constraint functions for CMA evolution strategies," in *Proceedings of the 32nd German Annual Conference on Artificial Intelligence (KI '09)*, pp. 169–176, Paderborn, Germany, September 2009.

[4] O. Kramer, S. Brugger, 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 '07)*, pp. 666–673, ACM Press, London, UK, July 2007.

[5] O. Kramer and H.-P. Schwefel, "On three new approaches to handle constraints within evolution strategies," *Natural Computing*, vol. 5, no. 4, pp. 363–385, 2006.

[6] O. Kramer, C.-K. Ting, and H. Kleine Büning, "A mutation operator for evolution strategies to handle constrained problems," in *Proceedings of the 7th Conference on Genetic and Evolutionary Computation (GECCO '05)*, pp. 917–918, Washington, DC, USA, June 2005.

[7] O. Kramer, C.-K. Ting, and H. Kleine Büning, "A new mutation operator for evolution strategies for constrained problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 2600–2606, Edinburgh, UK, September 2005.

[8] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, Germany, 1973.

[9] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen Mittel der Evolutionsstrategie*, Birkhäuser, Basel, Switzerland, 1977.

[10] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, 2002.

[11] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[12] J. C. Bean and A. B. Hadj-Alouane, "A dual genetic algorithmfor bounded integer programs," Tech. Rep., University of Michigan, Kalamazoo, Mich, USA, 1992.

[13] A. Fiacco and G. McCormick, "The sequential unconstrained minimization technique for nonlinear programming—a primal-dual method," *Management Science*, vol. 10, pp. 360–366, 1964.

[14] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[15] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, D. B. Fogel, Ed., pp. 579–584, IEEE Press, Orlando, Fla, USA, June 1994.

[16] 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 '98)*, pp. 518–522, Mainz, Aachen, Germany, September 1998.

[17] R. G. L. Riche, C. Knopf-Lenoir, and R. T. Haftka, "A segregated genetic algorithm for constrained structural optimization," in *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA '95)*, L. J. Eshelman, Ed., pp. 558–565, University of Pittsburgh, Morgan Kaufmann, San Francisco, Calif, USA, July 1995.

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

[18] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[19] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, pp. 971–978, 2001.

[20] D. Powell and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 424–431, University of Illinois at Urbana-Champaign, Morgan Kaufmann, San Francisco, Calif, USA, July 1993.

[21] F. Hoffmeister and J. Sprave, "Problem-independent handling of constraints by use of metric penalty functions," in *Proceedings of the 5th Conference on Evolutionary Programming (EP '96)*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds., pp. 289–294, MIT Press, Cambridge, UK, February 1996.

[22] A. I. Oyman, K. Deb, and H.-G. Beyer, "An alternative constraint handling method for evolution strategies," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 1, pp. 612–619, IEEE Service Center, Piscataway, NJ, USA, July 1999.

[23] T. P. Runarsson, "Approximate evolution strategy using stochastic ranking," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 2760–2767, IEEE, Vancouver, Canada, July 2006.

[24] S. V. Belur, "CORE: constrained optimization by randomevolution," in *Proceedings of the Late Breaking Papers at the Genetic Programming Conference*, J. R. Koza, Ed., pp. 280–286, Stanford University, Stanford, Calif, USA, July 1997.

[25] 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*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[26] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

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

[28] J. Paredis, "Co-evolutionary constraint satisfaction," in *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature (PPSN '94)*, pp. 46–55, Springer, Jerusalem, Israel, October 1994.

[29] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature (PPSN '96)*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., pp. 245–254, Berlin, Germany, September 1996.

[30] C. A. Coello Coello, "Constraint handling through a multiobjective optimization technique," in *Proceedings of the Genetic and Evolutionary Computation Conference*, A. S. Wu, Ed., pp. 117–118, Orlando, Fla, USA, July 1999.

[31] C. A. Coello Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Engineering Optimization*, vol. 32, no. 3, pp. 275–308, 2000.

[32] F. Jimenez and J. L. Verdegay, "Evolutionary techniques for constrained optimization problem," in *Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99)*, H.-J. Zimmermann, Ed., Mainz, Aachen, Germany, September 1999.

[33] E. Mezura-Montes and C. A. Coello Coello, "Constrained optimization via multiobjective evolutionary algorithms," *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pp. 53–75, 2008.

[34] I. C. Parmee and G. Purchase, "The development of a directed genetic search technique for heavily constrained design spaces," in *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control (PEDC '94)*, I. C. Parmee, Ed., pp. 97–102, University of Plymouth, Plymouth, UK, September 1994.

[35] M. Schoenauer and S. Xanthakis, "Constrained GA optimization," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 573–580, Morgan Kaufman, San Francisco, Calif, USA, July 1993.

[36] P. D. Surry, N. J. Radcliffe, and I. D. Boyd, "Amulti-objective approach to constrained optimisation of gas supply networks: the COMOGA Method," in *Proceedings of the Evolutionary Computing, AISB Workshop*, T. C. Fogarty, Ed., Lecture Notes in Computer Science, pp. 166–180, Springer, Sheffield, UK, April 1995.

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

[38] T. Takahama and S. Sakai, "Constrained optimization by the e constrained differential evolution with gradient-based mutation and feasible elites," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 1–8, IEEE Press, Vancouver, Canada, July 2006.

[39] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Proceedings of the IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 9–16, IEEE Press, Vancouver, Canada, July 2006.

[40] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, G. G. Yen, S. M. Lucas, G. Fogel, et al., Eds., pp. 25–32, Vancouver, Canada, July 2006.

[41] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Computation*, vol. 198 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 2009.

[42] H.-P. Schwefel, *Evolutionsstrategie und numerische optimierung*, Ph.D. thesis, TU Berlin, Berlin, Germany, 1975.

[43] H.-P. Schwefel, *Evolution and Optimum Seeking. Sixth-Generation Computer Technology*, Wiley Interscience, New York, NY, USA, 1995.

[44] D. V. Arnold and D. Brauer, "On the behaviour of the (1+1)-es for a simple constrained problem," in *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature (PPSN '08)*, pp. 1–10, Dortmund, Germany, September 2008.

[45] H.-P. Schwefel, "Adaptive mechanismen in der biologischen evolution und ihr einfluss auf die evolutionsgeschwindigkeit," in *Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik*, TU Berlin, Berlin, Germany, July 1974.

[46] N. Hansen, "The cma evolution strategy: a tutorial," Tech. Rep., TU Berlin, ETH Zürich, Germany, 2005.

[47] A. Ostermeier, A. Gawelczyk, and N. Hansen, "A derandomized approach to self adaptation of evolution strategies," *Evolutionary Computation*, vol. 2, no. 4, pp. 369–380, 1994.

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

[48] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel-assisted evolution strategies," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN '02)*, pp. 361–370, Granada, Spain, September 2002.

[49] S. Kern, N. Hansen, and P. Koumoutsakos, "Local metamodels for optimization using evolution strategies," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06)*, pp. 939–948, Reykjavik, Iceland, 2006.

[50] H. Ulmer, F. Streichert, and A. Zell, *Optimization by Gaussian Processes Assisted Evolution Strategies*, Springer, Heidelberg, Germany, 2003.

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

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, Berlin, Germany, 2009.

4795, 2010, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/2010/185063 by Cochrane Netherlands, Wiley Online Library on [24/06/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License