# An Alternative Constraint Handling Method for Evolution Strategies

**Ahmet İrfan Oyman**
University of Dortmund
Department of Computer Science XI
Systems Analysis Research Group
44221 Dortmund, Germany
oyman@LS11.cs.uni-dortmund.de

**Kalyanmoy Deb**
Indian Institute of Technology
Department of Mechanical Engg
Kanpur GA Lab (KanGAL)
Kanpur, PIN 208 016, India
deb@iitk.ac.in

**Hans-Georg Beyer**
University of Dortmund
Department of Computer Science XI
Systems Analysis Research Group
44221 Dortmund, Germany
beyer@LS11.cs.uni-dortmund.de

**Abstract- Most real-world search and optimization problems are faced with constraints, which must be satisfied by any acceptable solution. Although a plethora of research is spent on handling constraints in genetic algorithms (GAs), the same is not the case in evolution strategies (ESs). However, this does not say that ESs have not been applied to real-world problems. In fact, in the absence of an efficient constraint-handling technique, ES practitioners have mostly made sure that their ESs started from feasible solutions, a matter which allowed them to apply the commonly-used rejection scheme. In this paper, we borrow a constraint-handling scheme from the GA literature and implement it with standard ES paradigm. The resulting algorithm does not require initial feasible solutions and is found to yield a faster progress in the cylindrical corridor model and be efficient in solving a couple of complicated test problems. The results are interesting and suggest further use of the proposed technique in real-world search and optimization problems.**

## 1 Introduction

Several optimization problems are defined by a fitness function to be minimized (or maximized) and by a set of constraints which should be fulfilled. In such problems, the most important two aims are locating the feasible region and finding the point therein yielding the best fitness value. Therefore, one would prefer an algorithm which can fulfill both of these.

The original and standard method of handling constraints in ES algorithms is the rejection scheme [Sch75, Sch95]. A significant improvement has been attained by the metric penalty function (MPF) method proposed by Hoffmeister and Sprave [HS96]. This method is reported to perform well on the (rectangular) corridor model. It optionally uses a metric based on the distance to the closest feasible point for infeasible individuals. This metric defined in the search space is reported to be tedious to evaluate by the authors, and omitted in the simulations. Therefore, another metric based on the worst attainable fitness value in the feasible region is used alone in the selection operator. This second metric may also be difficult to be determined depending on the problem considered. Anyway, the MPF method was successfully used in constrained optimization of real-world problems, e.g. [SS96]. For a survey on constrained optimization using evolutionary algorithms, see Michalewicz and Schoenauer [MS96].

This paper uses a penalty function approach proposed for GAs by the second author [Deb98]. This method uses two metrics. One is directly based on violated constraints obtained by simple addition and the other on the worst fitness value of feasible individuals entering the selection operator in the current generation. It works with substitute fitness values for infeasible individuals and actual fitness values of feasible individuals. Therefore, it has a very simple selection operator. These two metrics used are similar in nature to the metrics proposed in [HS96]. However, they are much easier to compute, particularly if they are measured in the context of the individuals at a given generation. This method is applied in this paper to different ES algorithms for the first time. The results are compared with the ones attained by the rejection method.

The paper is organized as follows: Section 2 introduces the constraint handling methods used: the rejection scheme and the dynamic update scheme. It further includes the descriptions of the ES algorithms used, especially the $(1+1)$-ES and the $(\mu/\mu_I+\lambda)$-ES. The three test problems used in the comparison are presented in Section 3. Thereafter, the results are summarized in Section 4. Section 5 provides conclusions and outlines possible research directions and the work to be done in the future.

## 2 Algorithms

The algorithms used in this paper are introduced in this section. The $(1+1)$-ES is used in most simulations; therefore, it will be explained explicitly. Additionally, the algorithms $(1+10)$-ES, $(2+10)$-ES, and $(2/2_I+10)$-ES are used. These three algorithms are the special cases of $(1+\lambda)$-ES, $(\mu+\lambda)$-ES, and $(\mu/\mu_I+\lambda)$-ES, respectively. The symbol $\lambda$ stands for the number of descendants created per generation, and $\mu$ for the number of parents. The plus sign indicates that *elitist (truncation) selection* is used. The $(\mu/\mu_I+\lambda)$-ES additionally uses the intermediate recombination operator (also called intermediary). More information on evolution strategies can be found in the literature [Rec73, Rec94, Sch95, BFM97].

The $(1+\lambda)$-ES is a special case of the $(\mu+\lambda)$-ES for $\mu=1$, and the $(1+1)$-ES is a special case of the $(1+\lambda)$-ES for $\lambda=1$, respectively. In all cases, the ES algorithm starts at a given (or randomly selected) point in the search space. If $\mu > 1$, $\mu$ different starting points can be used. In each generation, $\lambda$ descendants are generated from the $\mu$ parents. For the $(\mu+\lambda)$-ES, the parent for each descendant is selected using a uniform random distribution. The normal distribution $\mathcal{N}(0, \sigma^2)$ is used in the creation operation (called *mutation*). In this paper, the same standard deviation $\sigma$ is used for all $N$ object variables. The selection operator selects $\mu$ individuals having the best fitness values among the $\mu$ parents and $\lambda$ newly generated descendants as the parents of the next generation (called *truncation selection*). That is, one selects the individuals with smaller fitness values in the minimization case and larger ones in the maximization case. The generation loop of selecting the parent, mutation, fitness evaluation, and truncation selection continues until the termination criterion is fulfilled.

The $(1+1)$-ES is shown in Algorithm 1. It is the simplest evolutionary algorithm. It starts with the evaluation of the fitness of the initial parent. The generation loop is evaluated until the termination criterion is fulfilled. The selection operator may select the descendant if it is at least as good as its parent (denoted as the "$\leq$" option for the minimization case) or may force it to be strictly better than the parental individual for this replacement (the "$<$" option, the default case in this paper). The termination criterion used in this work is the evaluation of the fitness values for a given number of descendants or obtaining a near-optimal fitness value, whichever occurs the first. The near-optimality is defined as the closeness of the obtained objective function value from the optimum value.

The $(\mu/\mu_{\mathrm{I}} + \lambda)$-ES differs from the $(\mu + \lambda)$-ES by additionally implementing the recombination operator. It is shown in Algorithm 2. The function *centroid*($\mathbf{P}^{(g)}$) returns the center of mass of $\mu$ parents, which will be used in the next step to generate mutations from. It can actually be executed just once before the $l$-loop, since $\mathbf{E}_l$ is independent of $l$. For the $(\mu + \lambda)$-ES, however, the operator *mate*($\mathbf{P}^{(g)}$) is used instead in the $l$-loop, which will select one of the parents using a uniform random distribution.

Some revisions must be carried out on these algorithms if the fitness cannot be expressed by a scalar value alone. For instance, constraints must be fulfilled in case of constrained optimization. If constraints are not fulfilled, the objective function value is usually not meaningful. Two possibilities considered in this paper are introduced below.

**algorithm** $(1+1)$-ES
**begin**
  $g := 0$
  *initialize*($\mathrm{P}^{(0)}$) $\left[\mathrm{P}^{(0)} := \left(\mathbf{x}_{\mathrm{P}}^{(0)}, F(\mathbf{x}_{\mathrm{P}}^{(0)})\right)\right]$
  **while** **not** *terminate*() **do**
    $\mathbf{x}_{\mathrm{C}} :=$ *mutate*($\mathbf{x}_{\mathrm{P}}^{(g)}, \sigma$)
    $F_{\mathrm{C}} := F(\mathbf{x}_{\mathrm{C}})$
    $\mathrm{C} := (\mathbf{x}_{\mathrm{C}}, F_{\mathrm{C}})$
    $\mathrm{P}^{(g+1)} :=$ *select*($\mathrm{C}, \mathrm{P}^{(g)}$)
    $g := g + 1$
  **od**
**end**

Algorithm 1: The $(1+1)$-ES algorithm.

**algorithm** $(\mu/\mu_{\mathrm{I}} + \lambda)$-ES
**begin**
  $g := 0$
  *initialize*($\mathbf{P}^{(0)}$)
    $\left[\mathbf{P}^{(0)} := \left\{\left(\mathbf{x}_m^{(0)}, F(\mathbf{x}_m^{(0)})\right),\ m = 1, \ldots, \mu\right\}\right]$
  **while** **not** *terminate*() **do**
    **for** $l := 1$ **to** $\lambda$ **do**
      $\mathbf{E}_l :=$ *centroid*($\mathbf{P}^{(g)}$)
      $\tilde{\mathbf{x}}_l :=$ *mutate*($\mathbf{E}_l, \sigma$)
      $\tilde{F}_l := F(\tilde{\mathbf{x}}_l)$
    **od**
    $\tilde{\mathbf{P}}^{(g)} := \left\{(\tilde{\mathbf{x}}_l, \tilde{F}_l),\ l = 1, \ldots, \lambda\right\}$
    $\mathbf{P}^{(g+1)} :=$ *select*($\tilde{\mathbf{P}}^{(g)}, \mathbf{P}^{(g)}$)
    $g := g + 1$
  **od**
**end**

Algorithm 2: The $(\mu/\mu_{\mathrm{I}} + \lambda)$-ES algorithm.

**The rejection scheme:** In this scheme, the loop of generating descendants is executed until $\lambda$ feasible individuals are found [Sch95, Appendix B3]. All other infeasible individuals generated in the mean time are rejected. That is, the number of individuals created in a generation (denoted as $\lambda_{actual}$), and therefore the number of fitness evaluations, is unknown and changes from generation to generation.

Definitely, $\lambda_{actual}$ depends on many things: The fitness function, where the parents $\mathbf{P}^{(g)}$ are in the search space, the shape of the feasible region, the mutation strength, etc. This scheme uses only the fitness values in the selection operator. If all parents are initialized to the *infeasible* region, the algorithms with "$\leq$" option carry out a random search until at least one feasible solution is found. The algorithms with "$<$" option continue to generate individuals from the same point(s) until $\lambda$ feasible descendants are found.

**The dynamic update scheme:** The dynamic update scheme is implemented just before the selection operation. It uses the constraint violation information obtained from $\lambda$ descendants generated and from the parents in order to guide the search toward the feasible region. The proposed method is similar to that suggested by Deb [Deb98] for genetic algorithms. The method is also similar to that suggested by Hoffmeister and Sprave [HS96], however, here we use a fitness function which depends on the parent and children population at every generation and, therefore, becomes a dynamic approach.

This goal is pursued in two steps: First of all, the fitness value of the worst feasible individual entering the selection operator is determined. This quantity is called $f_{worst}$. If all individuals are infeasible, one has $f_{worst} = 0$. The second step aims to guarantee that all infeasible individuals entering the selection operator have fitness values worse than $f_{worst}$. Since the constraints are written in $g_i(\mathbf{x}) \geq 0$ form, the violated constraints will give negative values. These negative values are summed over all constraints for an individual and the absolute value of this sum is used as a *heuristic* measure on the degree of violation. Each infeasible individual is assigned a fitness equal to the sum[1] of this measure and $f_{worst}$. However, each feasible solution is assigned a fitness equal to the solution's objective function value. Note that the objective function value of an infeasible solution is never computed, a matter which has a practical meaning [Deb98].

Thus, this scheme guarantees the following events:

1. A feasible solution always has a better fitness value than an infeasible solution,
2. Feasible solutions are assigned fitness according to the objective function value solely,
3. An infeasible solution with a small constraint violation measure is assigned better fitness than an infeasible solution with a large constraint violation measure, and
4. Infeasible solutions are assigned fitness only based on their constraint violation measure. This eliminates the need to have separate penalty parameters.

# 3 Test problems

Three problems are considered in the experiments carried out in this paper, namely the cylindrical corridor model, `test1`, and `test2`. The *cylindrical corridor model* [Sch95, p. 361]

$$\begin{array}{ll} \text{Maximize} & f_{cylinder}(\mathbf{x}) := c \cdot \mathbf{v}^{\mathrm{T}}\mathbf{x} \\ \text{Subject to} & g(\mathbf{x}) \equiv \quad b - \|(\mathbf{v}^{\mathrm{T}}\mathbf{x})\mathbf{v} - \mathbf{x}\| \geq 0 \end{array} \tag{1}$$

was introduced in the large problem catalog of [Sch75] for the first time ($c, b \in \mathrm{I\!R}^+$). It is presented in a different notation there. The vector $\mathbf{v}$ with $\|\mathbf{v}\| = 1$ gives the direction of the corridor in the search space. The special case $v_1 = 1$, $v_{i\neq 1} = 0$ is used in the simulations here, although its value does not affect the performance of ES algorithms considered. The constraint $g(\mathbf{x})$ defines the feasible region: The distance to the corridor axis should be less than or equal to $b$ for a feasible point. If one denotes the number of variables with $N$, this corresponds to the Euclidean distance in the $(N-1)$-dimensional subspace.

The other two test problems (`test1` and `test2`) occur in [Mic95] as "test case #3" and "test case #5", respectively. They are also investigated in [Deb98] under the names "test problem 5" and "test problem 8", respectively.

The test problem `test1` reads

$$\begin{array}{l} \text{Minimize} \\ f_1(\mathbf{x}) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 \\ \qquad +10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \end{array} \tag{2}$$

$$\begin{array}{l} \text{Subject to} \\ g_1(\mathbf{x}) \equiv 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0, \\ g_2(\mathbf{x}) \equiv 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0, \\ g_3(\mathbf{x}) \equiv 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0, \\ g_4(\mathbf{x}) \equiv -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0, \\ -10 \leq x_i \leq 10, \quad i = 1, \ldots, 7. \end{array} \tag{3}$$

The optimal solution is

$$\begin{aligned} \hat{\mathbf{x}}^{\mathrm{T}} = (2.330499,\ 1.951372,\ -0.4775414, \\ 4.365726,\ -0.6244870,\ 1.038131,\ 1.594227), \\ \hat{f}_1 = 680.6300573. \end{aligned}$$

At this solution, constraints $g_1$ and $g_4$ are active. Michalewicz [Mic95] reported that the feasible region for this problem occupies only about $0.5121\%$ of the search space.

The test problem `test2` reads

$$\begin{array}{l} \text{Minimize} \\ f_2(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 \\ \qquad +4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 \\ \qquad +7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45, \\ \text{Subject to} \\ g_1(\mathbf{x}) \equiv 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0, \\ g_2(\mathbf{x}) \equiv -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0, \\ g_3(\mathbf{x}) \equiv 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0, \\ g_4(\mathbf{x}) \equiv -3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0, \\ g_5(\mathbf{x}) \equiv -5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0, \\ g_6(\mathbf{x}) \equiv -x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0, \\ g_7(\mathbf{x}) \equiv -0.5(x_1-8)^2 - 2(x_2-4)^2 - 3x_5^2 + x_6 + 30 \geq 0, \\ g_8(\mathbf{x}) \equiv 3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0, \\ -10 \leq x_i \leq 10, \quad i = 1, \ldots, 10. \end{array} \tag{4}$$

The optimum solution to this problem is as follows:

$$\begin{aligned} \hat{\mathbf{x}}^{\mathrm{T}} = (2.171996,\ 2.363683,\ 8.773926,\ 5.095984,\ 0.9906548, \\ 1.430574,\ 1.321644,\ 9.828726,\ 8.280092,\ 8.375927), \\ \hat{f}_2 = 24.3062091. \end{aligned}$$

The first six constraints are active at this solution.

# 4 Experiments

## 4.1 How to measure performance

In the ES literature, the most important quantity for measuring the performance of the algorithms is the progress rate $\varphi$ [Rec73, Sch75, Sch95]. This quantity is defined as the expected decrease in the distance to the optimum in *one* generation. The Euclidean distance to the optimum is measured between the centroids of the generations and the optimum. In the experiments, the arithmetic mean of numerous experiments and the standard deviation of it (i.e. the standard error of the measured useful distances traveled) are used to estimate the theoretical quantity $\varphi$. The arithmetic mean and standard error pair are shortly written as "$mean \pm s.e.$".

To measure $\varphi$ *statically* (the usual way practiced for the investigations on the sphere model, e.g. [Bey96]), the measurements are taken at the same parental individual(s). The measurements are "One-Generation-Experiments" in this case. Such static measurements express local progress behavior; therefore, they must be repeated for all different characteristic variable settings of the fitness function. Depending on

[1] In the maximization problems, one has to subtract the constraint violation measure from $f_{worst}$.

the fitness function, this may be the whole search space in the worst case. Therefore, some specific test functions are chosen for the theoretical analysis so that the theoretical and empirical analysis gets simpler. For the sphere model, this set of variable settings reduces to a single point in the search space since the spherical symmetry and the normalization used enables the generalization of the obtained results to different values of $N$, $R$ (distance to the optimum), and $\sigma$.

The next stage in the analysis has been done on the ridge functions [OBS98, Oym98a]. In these functions, the set of variable settings required for the analysis is a one-dimensional manifold, since the values measured depend on the distance to the ridge axis. As a result, this distance occurs in the theoretical formulae. The measurements must be done for different values of it.

Similarly, the ES algorithms on the cylindrical corridor model attain different values of $\varphi$ depending on the distance $r$ to the corridor axis. This is also the case in the feasible region. For a rigorous analysis, the measurements should be taken statically for different values of $r$. Another way to measure the performance is the *dynamic* analysis: One can measure the overall performance of the algorithm by measuring the useful distance traveled over several consecutive generations. This makes sense if the distance $r$ takes a *stationary* value over generations; in other words, if the standard deviation of its arithmetic mean goes to zero as the number of measurements goes to infinity. This behavior can be observed on ridge functions. However, the result obtained on ridge functions cannot be used in the analysis of corridor models, nor they are necessary to understand the underlying paper. Preliminary simulations on the cylindrical corridor model indicate the dependence of $\varphi$ on $r$ especially for larger values of $\sigma$. They also show that the stationary $r$ value is slightly smaller than $b$ (see Equation 1).

The stationary performance of both constraint handling methods (the rejection scheme and the dynamic update scheme) can be compared if the measurements are taken after reaching the stationary value of $r$. Otherwise, the dynamic performance is measured. For the stationary case, the experiments are started on the corridor axis, and a transient period (denoted by $\nu_T$ fitness evaluations) is reserved before starting to collect data. This value suffices for the cases considered. Thereafter, the data is collected for the following $\nu$ fitness evaluations.

It is very important to note that the performance of the rejection scheme and of the dynamic update method *cannot* be compared using the $\varphi$ measure. The reason is simple: The number of actual fitness evaluations ($\lambda_{actual}$) *changes* for the rejection scheme from generation to generation, whereas one has $\lambda_{actual} = \lambda$ for the dynamic update scheme. Therefore, one has to measure the progress per *fitness evaluation* for a fair comparison. For each generation, the measured progress rate is divided by the value of $\lambda_{actual}$ for both constraint handling methods. To summarize, the measure

$$\phi := \text{expected progress per fitness evaluation} \qquad (5)$$

is used in this paper.

**Other metrics.** The measure $\phi$ is fair if one starts in the feasible region of the corridor model. If the starting point is *not* feasible, one has to use another metric: The number of fitness evaluations to reach the feasible region is used for the comparison, which depends on the initial distance $r^{(0)}$ to the corridor axis ($r^{(0)} > b$).

The test cases `test1` (2,3) and `test2` (4) do not have a nice corridor. The static analysis of $\phi$ would be too complex. Therefore, the number of generations to reach the vicinity of the optimal fitness value is used as the metric. The two algorithms compared are set to the same initial values of the objective function variables for a fair comparison. The measurements are repeated for several different initial variable settings. The "vicinity" is defined as a relative error less than three per cent (or 6% at some simulations) and that the best fitness value found yielding this error belongs to a feasible point.

**Other considerations.** The normally distributed pseudo-random values are generated using the Box-Muller method in Numerical Recipes [PTVF92, p. 289]. This algorithm uses uniformly distributed values; the procedure `ran2` [PTVF92, p. 282] is used to generate the pseudo-random variables required. Different copies of these pseudo-random generators are used for different stochastic processes: For the ES algorithms concerned, this was the case for the $(2 + 10)$-ES, since one needs a second independent generator for the selection of the parent which is used in the creation of the descendant. Different random seeds are used for these independent pseudo-random processes and for independent simulation runs.

### 4.2 Overview on the experiments

This section summarizes the simulation results obtained for the two constraint handling methods, the dynamic update scheme and the rejection scheme, respectively. The aim is to compare their performance on different test functions and for different ES algorithms.

The simulation runs are done on three test problems: The cylindrical corridor model, `test1`, and `test2`. For the cylindrical corridor, several ES algorithms are used where the search is initialized in the feasible region. ES with both dynamic and rejective scheme are compared with respect to the $\phi$ measure. Thereafter, the $(1 + 1)$-ES is considered in the infeasible region, and the number of fitness evaluations necessary to enter the feasible region is compared. The values $b = 450$ and $N = 100$ are chosen arbitrarily for the cylindrical corridor. On the problems `test1` and `test2`, the number of fitness evaluations necessary to get to the vicinity of the optimum fitness value is measured.

### 4.3 The feasible region of the cylindrical corridor

The first comparison of the dynamic update with the rejection scheme is done in the feasible region of the cylindrical corridor model. Four different ES algorithms are used in this comparison: The $(1+1)$-ES, the $(1+10)$-ES, the $(2+10)$-ES, and the $(2/2_{\mathrm{I}}+10)$-ES. The results are shown in the Figures 1–4, respectively. The "$<$" version of these algorithms are used in the comparison; i.e. the descendant *cannot* replace the parent if their fitness values are exactly the same. However, the results do not change much if this replacement would be allowed, since this equivalence case is rare in the feasible region.

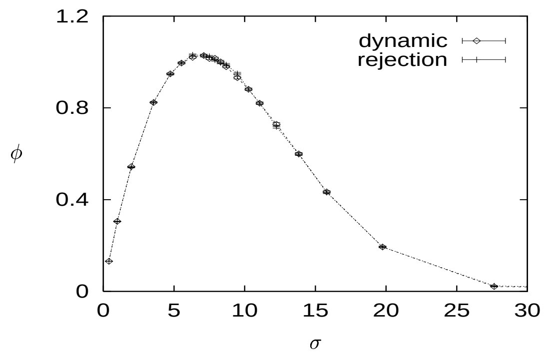


Figure 1: $(1+1)$-ES, $\nu_T = 2\,000$, $\nu = 10\,000$.
ratio: $1.028/1.031 \approx 0.998$.

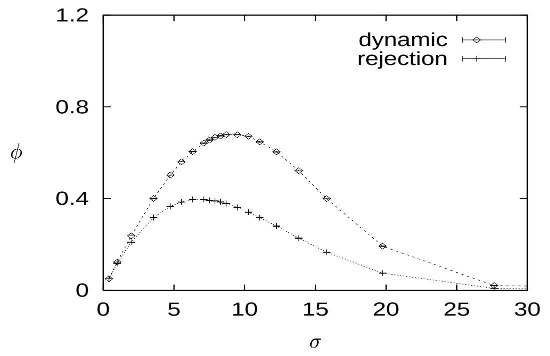


Figure 2: $(1+10)$-ES, $\nu_T = 10\,000$, $\nu = 50\,000$.
ratio: $0.679/0.397 \approx 1.71$.

40 independent runs are made for each result shown. The average and standard error of the measured $\phi$ is shown for different values of $\sigma$. The number of fitness evaluations per simulation run is shown in the caption of the figures, as well as the ratio of the optimal $\phi$ values (denoted as $\hat{\phi}$) for these two algorithms. This ratio is obtained by dividing the $\hat{\phi}$ value for the dynamic update scheme by the corresponding value for the rejection scheme.

One observes that these two constraint handling methods show the same performance for the $(1+1)$-ES. This is obvious since the dynamic update scheme does not differ for this algorithm from the rejection scheme if the parent is feasible. These schemes are also equivalent for the $(\mu+1)$-ES operating in the feasible region (since infeasible descendants cannot become parents anymore in the dynamic update scheme if once all parents are feasible). This so-called steady state ES algorithm is not considered in this paper.

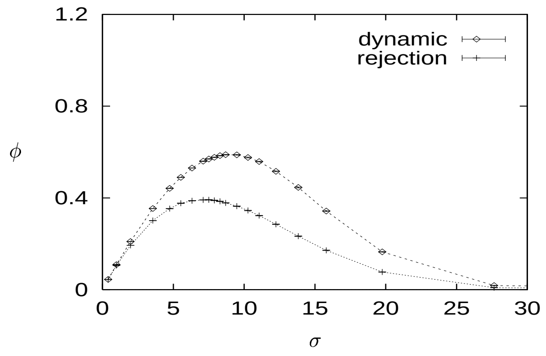


Figure 3: $(2+10)$-ES, $\nu_T = 10\,000$, $\nu = 50\,000$.
ratio: $0.588/0.392 \approx 1.50$.

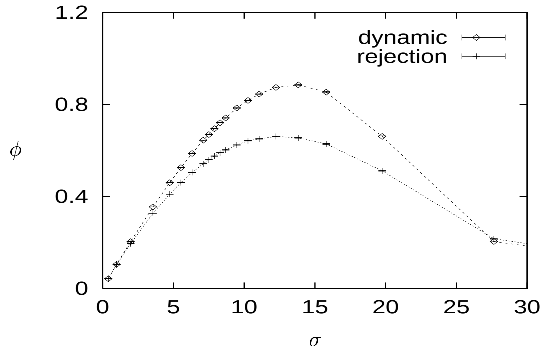


Figure 4: $(2/2_{\mathrm{I}}+10)$-ES, $\nu_T = 10\,000$, $\nu = 50\,000$.
ratio: $0.886/0.661 \approx 1.34$.

A difference in the performance is observed for the other three cases with $\lambda > 1$. The dynamic update scheme does not wait until $\lambda$ feasible solutions are generated. The feasible descendants become parents as long as they are feasible and have a fitness value better than a parent which they can substitute.

The maximum $\phi$ is obtained for the $(1+1)$-ES among the algorithms considered. This algorithm is shown to be the most efficient one for the sphere model [Bey96]. This has been observed also for ridge functions [Oym98b]. These results should be considered carefully if one would make a generalization: The $(1+1)$-ES is by no means guaranteed to be the best optimization algorithm. The results may change considerably in multimodal fitness landscapes. However, it is clear that for a $(\mu+\lambda)$-ES, if all $\lambda$ children solutions are either feasible or infeasible, there is no distinction between the two schemes. The interesting scenario emerges when some children solutions are feasible and some are infeasible. Such a scenario is most likely to happen in complicated problems; an ES with rejection scheme will keep on creating solutions till $\lambda$ feasible solutions will be found. Whereas, an ES with the dynamic update scheme can progress with fewer feasible solutions. Thus, it is also somewhat intuitive that in such a scenario, an ES with the dynamic update scheme will perform better than with the rejection scheme.

Therefore, because of the *simplicity* of the $(1+1)$-ES, and because of the equivalence in the performance of the two constraint handling schemes with the $(1+1)$-ES in the feasible region, the experiments are carried out only for the $(1+1)$-ES in the rest of this paper. Before continuing with other experiments, one has to note several other simple observations in Figures 1–4. First of all, the optimal value of $\sigma$ yielding $\hat{\phi}$ (denoted as $\hat{\sigma}$) is *different* for the two constraint handling schemes (of course, except for the $(1+1)$-ES). One observes that $\phi$ decreases if $\mu$ is increased without recombination (compare Figure 2 with Figure 3), and increases if recombination is used (compare Figure 3 with Figure 4). Moreover, recombination causes an increase in $\hat{\sigma}$. These results are in accordance to the ones from the theory obtained on the sphere model and on ridge functions.

### 4.4 The infeasible region of the cylindrical corridor

The observations in the previous subsection showed that the performance of both constraint handling methods become the same for the $(1+1)$-ES if one starts in the *feasible* region (Figure 1). The dynamic observations on the infeasible region will complement this result. For this purpose, the initial distance $r^{(0)}$ to the corridor axis is set to different values, and the number of fitness evaluations required to get into the feasible region is measured. Since the feasible region may not be known a priori, finding the feasible region can be considered as an important task of any optimization algorithm.

For the comparison, the $(1+1)$-ES with "$<$" option is used. The "$\leq$" option would result in random search for the rejection scheme, since the fitness value is considered as undefined or "$-\infty$" in the infeasible region. We measured the number of fitness evaluations arbitrarily for $\sigma = 2$ and for 40 independent simulations until a feasible descendant is obtained. The same set of random numbers and the same set of initial variable settings are used in comparing the two constraint handling methods. The simulation is terminated as unsuccessful if the feasible region is not reached after one million fitness evaluations (denoted as $\nu = 1\,000\,000$). The simulation is repeated for the rejection scheme with $\sigma = 4$ to obtain a better interpretation. The results are summarized in two figures.

Figure 5 shows the number of unsuccessful runs for both constraint handling methods. The performance of the rejection scheme strongly depends on $r^{(0)}$, and slightly on $\sigma$. For larger values of $r^{(0)}$, the probability to get into the feasible region gets smaller. This figure shows the qualitative difference between the two constraint handling methods. A change in the performance of the dynamic update scheme is not observed for these $r^{(0)}$ values. The dependence on $\sigma$ would be relatively negligible.

For the qualitative comparison, one needs the average number of fitness evaluations for successful simulation runs.

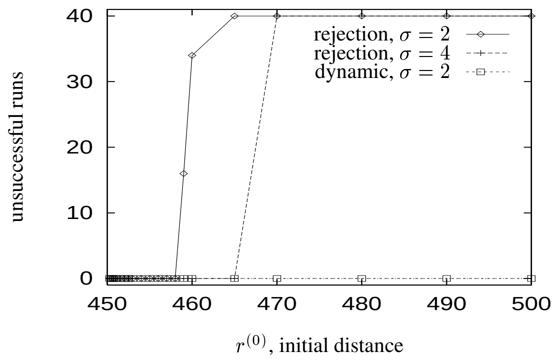


Figure 5: Cylindrical corridor model, $(1+1)$-ES, number of unsuccessful simulations

The averages and standard errors of the results are shown in Figure 6. The vertical axis is chosen to be logarithmic to demonstrate the differences more clearly. One observes a linear increase for the dynamic update scheme, and an (over)-exponential increase for the rejection scheme.

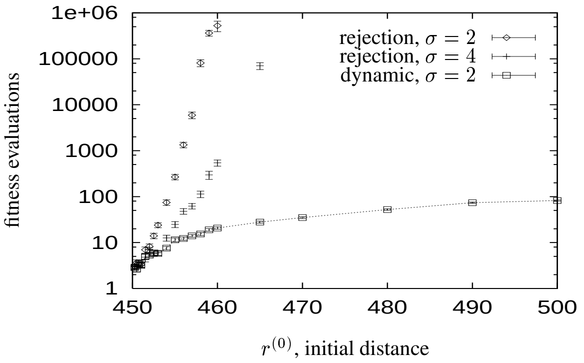


Figure 6: Cylindrical corridor model, $(1+1)$-ES, number of fitness evaluations

The dynamic update scheme uses the constraint violation information to guide the search toward the feasible region. The algorithm moves over generations closer to this region after each successful descendant: Any descendant with smaller violation value substitutes its parent. Therefore, the problem of locating the feasible region reduces to the problem of finding individuals which are closer to the feasible region. The progress rate values for the $(1+1)$-ES on the sphere model can be used to estimate the number of fitness evaluations necessary to get into the feasible region. Such analysis can be found in [Rec73, Bey96].

The rejection scheme has to get into the feasible region in a single step. This step is easy if $r^{(0)}$ is only slightly larger than $b$; however, it is extremely hard if $r^{(0)}$ is much larger. For some values of $r^{(0)}$ and $\sigma$, the results of both constraint handling schemes are comparable, but in general the rejection scheme requires more fitness evaluations. The simulations of the rejection scheme for the $(1+1)$-ES with "$\leq$" option have not been done. However, they are expected to be similar to the ones with the random search method, since it does not use the constraint violation values as a quantitative metric to guide the search.

### 4.5 Test problem `test1`

The test problem `test1` is used to compare the performance of the two constraint handling schemes on the $(1+1)$-ES with "$<$" option. For the test functions `test1` and `test2`, the variables of the initial parent are initialized randomly between $-10$ and $+10$. The same set of initial parental variables is used in the simulations for the rejection scheme and the dynamic update scheme. The $(1+1)$-ES with "$\leq$" option would result in random search; however, it would probably leave the initialization interval.

Two simulation sets are carried out for both schemes, for two batch runs with 50 and 100 independent simulations, respectively. The simulations are terminated when the relative error of the best feasible solution found as compared to the optimum fitness value is less than three per cent (3%). The simulations with the dynamic update method ended successfully in less than $\nu = 3\,000$ fitness evaluations for $\sigma = 0.1$. The average of fitness evaluations over 50 simulations was 1 032, with a standard error of 48 (which will be denoted as $1\,032 \pm 48$), with a minimum of 331 and a maximum of 1 841.

The number of maximal allowed fitness evaluations is set to $\nu = 3\,000$. Thereafter, the performance of the dynamic update scheme is measured for $\sigma = 1$ and $\sigma = 5$. For such large values of the mutation strength, the number of successful simulations decreased. In the former case, 33 of 50 simulations were successful, whereas none was successful in the latter case.

The same simulation is repeated for the rejection scheme with the same set of initial points. The value of the mutation strength which gave the best results for the dynamic update scheme is used ($\sigma = 0.1$). The maximum number of fitness evaluations is increased to $\nu = 300\,000$, so that hundred times more fitness evaluations are available. The $(1+1)$-ES with "$<$" option could not locate the feasible region, which means that no movement in the search space was possible using the rejection scheme. For $\sigma = 2$, one out of 50 runs was able to come to 3% vicinity, in all other 49 cases 300 000 fitness evaluations have been exhausted. The successful run had 28 861 fitness evaluations, and 13 movements in the search space. In 29 cases of 50 runs, no movement in the search space was possible. In the 21 remaining cases, the feasible region was found; in four of them the best fitness found was in 6% vicinity, one of them being within 3%.

These results for the rejection scheme are not surprising in some sense since only $0.5121\%$ of the search space ($-10 \leq x_i \leq +10$) is feasible for this test problem [Deb98, Mic95]. This means that by randomly creating solutions in the search space, it would take on average about 195 fitness evaluations to find one feasible solution. Thus, with 300 000 fitness evaluations, a random search would have been found about 1 536 feasible solutions. The fact that an ES with $\sigma = 0.1$ could not even find one feasible solution in 300 000 fitness evaluations reveals how important it is to use a correct mutation strength in handling constrained optimization problems using an ES. We emphasize that locating the feasible region is the real problem in this case, since both constraint satisfaction algorithms behave exactly the same if they would start in the feasible region, or if they have reached it.

The simulation is repeated for $\sigma = 0.1$ hundred times on 100 different initial points for $\nu = 350\,000$ fitness evaluations. The rejection scheme could not locate the feasible region. The simulation run for the dynamic update scheme reached the 3% vicinity in all 100 cases, and had $1\,008.3 \pm 28$ for the average number of fitness evaluations, with the minimum 315 and maximum 1 912.

### 4.6 Test problem `test2`

The test problem `test2` is analyzed similar to `test1`. This problem is harder than the previous problem because the feasible region here is only 0.0003% of the entire search space [Mic95]. The $(1+1)$-ES with "$<$" option is investigated for $\nu = 300\,000$ fitness evaluations, Best results are obtained for $\sigma = 0.05$ for the dynamic update scheme. For 50 runs, the result was in the 3% vicinity for 33 runs, with the average number of fitness evaluations $182\,145 \pm 16\,956$, minimum 28 754, maximum 339 511. 15 out of 17 "unsuccessful" simulations were able to reach 6% vicinity of the optimum fitness value.

After repeating the simulation with the dynamic update scheme for $\nu = 350\,000$ and 100 runs, we found that 36 of the simulations were unsuccessful (although being within 6% vicinity), and the average of the fitness evaluations for the 64 successful ones (i.e. in the 3% vicinity) was $152\,283 \pm 10\,261$, minimum 23 410, maximum 330 491. If 6% vicinity is used as the termination condition, *all* 100 runs were successful, with the mean of the number of fitness evaluations $71\,924 \pm 5\,652$, minimum 10 040, maximum 286 865. Therefore, the problem is actually not hard for this latter termination condition.

Finally, the rejection method is considered under the same conditions (mutation strength, random seeds, set of initial variable settings, 3% vicinity as the termination condition). The feasible region could not be found for both cases with 50 and 100 runs. This is somewhat expected because with only 0.0003% feasible search region, a random search method will find only one feasible solution (on the average) in 350 000 fitness evaluations. After repeating the simulation for $\sigma = 2$, which is much larger than $\sigma = 0.05$, one out of 100 simulations was able to locate the feasible region, and the best fitness value found (1 339.31) was far away from the optimum value (24.31). As a result, one can say that the rejection scheme is not appropriate if the population is initialized in an infeasible region.

## 5 Conclusions & Outlook

Constraint handling is an important task of any search and optimization algorithm seeking to solve real-world problems. Although there exists a number of techniques for handling constraints in genetic algorithms (GAs), this issue has not been investigated enough in the context of evolution strategies (ESs). In this paper, we borrow one such technique from the GA literature and suggest a constraint handling technique for ESs. The suggested technique has been found to be better compared to the rejection scheme (commonly-used in ES studies) on the cylindrical corridor model and on two complex test problems. Although results were mainly presented with non-recombinative ESs, the suggested technique seems to be a promising effective tool for the use of multi-parent or recombinative ESs in solving constrained optimization problems.

This paper showed that the ES algorithms using the dynamic update method were able to progress toward the feasible region from infeasible regions. Moreover, in the feasible region, it attained in general larger progress rates per fitness evaluation as compared to the rejection method. These results were expected, since the dynamic update method uses the information provided by the constraints. Additionally, it has a definite advantage if $\lambda > 1$, since the parental set can move toward the regions with better fitness values at each single fitness evaluation.

The performance of both constraint handling methods considered here has not been compared yet on non-elitist ES algorithms, e.g. the $(1, \lambda)$-ES and the $(\mu, \lambda)$-ES.

The dynamic update method *itself* can be upgraded so that the value $f_{worst}$ is obtained using the second (or third) worst feasible individual. This upgrade will enable the selection of infeasible individuals as parents even if more than $\mu$ feasible individuals are available for the selection operator. This makes the operation of the ES algorithm on the boundaries of the feasible region(s) easier, probably yielding a speedup if the optimum is far away. It further allows a better focusing to an optimum with active constraints.

Finally, the self-adaptation of the mutation strength and deterministic schemes for controlling it were not applied at all in combination with the dynamic update scheme. As already known, the optimum value of the mutation strength *depends* on where the population is in the search space. Using the self-adaptation operator, optimum mutation strengths may be found adaptively and there seems no reason why the standard self-adaptive schemes will not work together with the proposed constraint handling scheme.

## 6 Acknowledgments

The first author thanks the DAAD (German Academic Exchange Service) for a PhD scholarship (grant # A/95/11445), his work is currently supported by the DFG, grant Be 1578/4-1. The second author is currently visiting the Systems Analysis Research Group of the Department of Computer Science at the University of Dortmund as an Alexander von Humboldt fellow. The third author is Heisenberg fellow of the DFG (German Research Foundation) under grant Be 1578/4-1.

## Bibliography

[Bey96] H.-G. Beyer. *Zur Analyse der Evolutionsstrategien*. Habilitationsschrift. University of Dortmund, Department of Computer Science, 1996.

[BFM97] Th. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[Deb98] K. Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 1998. in print.

[HS96] F. Hoffmeister and J. Sprave. Problem–independent handling of constraints by use of metric penalty functions. In *Evolutionary Programming V, ICEC'96*, 1996.

[Mic95] Z. Michalewicz. Genetic algorithms, numerical optimization and constraints. In *Genetic Algorithms: Proceedings of the 6th International Conference*, 1995.

[MS96] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[OBS98] A. I. Oyman, H.-G. Beyer, and H.-P. Schwefel. Where Elitists Start Limping: Evolution Strategies at Ridge Functions. In *Parallel Problem Solving from Nature - V*, 1998. http://ls11-www.cs.uni-dortmund.de/~oyman/TR/ppsn5.ps.gz.

[Oym98a] A. I. Oyman. *Convergence Behavior of Evolution Strategies on Ridge Functions*. Ph.D. Thesis, University of Dortmund, Department of Computer Science, 1998.

[Oym98b] A. I. Oyman. The performance of the $(1 + 1)$-ES on ridge functions. Unpublished manuscript, 1998.

[PTVF92] W.H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Verlag Frommann-Holzboog, 1973.

[Rec94] I. Rechenberg. *Evolutionsstrategie'94*. Frommann–Holzboog, Stuttgart, 1994.

[Sch75] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. Ph.D. Thesis, Technical University of Berlin, Department of Process Engineering, 1975.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[SS96] Martin Schütz and Joachim Sprave. Application of parallel mixed-integer evolution strategies with mutation rate pooling. In *Evolutionary Programming V, ICEC'96*, 1996.